DUBLIN INSTITUTE of TECHNOLOGY
Institiúid Teicneolaíochta Bhaile Átha Cliath

Small Scale Business Management Application

Jason Etienne / D19125729

Ciaran Cawley

A project submitted in partial fulfilment of the requirements of Dublin Institute of Technology for the degree of
B.Sc. (Ordinary) in Information Systems / Information Technology

April 2023

I certify that this project which I now submit for examination for the award of B.Sc. (Ordinary) in Information Systems / Information Technology, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: [illegible]

Date: ***02/04/2023***

1 ABSTRACT

This document details the Small Scale Business Management Application.

The intention of the application is to provide a user with a straightforward tool in order to help manage and streamline the operation of their business, such as the inventory tracking, booking scheduling and preparedness and financial overview.

The application is aimed towards self-employed trade people, such as painters and decorators, electricians and plumbers.

When researching I was unable to find a similar application as most I was unable to find any aimed at the smaller scale business I was intending to develop the application for as they seemed to be focused on enterprise level business.

Initial system design and requirements were done and then expanded and refined upon as development proceeded, these initial system design and requirements were of great benefit to guide the development to its final goals.

Some of the challenges encountered during development were around the Calendar application and Financial application. While I had some experience building calendar applications previously they were of a more basic nature and I found challenges in populated form data from saved bookings, also with the Financial application I had some experience but found the manipulation of data to then be depicted graphically a challenge and both were able to be overcome after researching.

There are some parts I would like to improve upon such as UI/UX and an issue I found during testing related to user registration, which I will discuss in the Conclusions section.

Overall I am happy with the final application as it accomplishes the goals I set out for it.

ACKNOWLEDGEMENTS

I would like to thank Ciaran Cawley. Ciaran was very helpful in pointing out enhancements or possible issues during development and suggesting ways of addressing them.
I found it extremely beneficial to have Ciaran’s guidance, suggestions and perspective during planning and development
I would also like to thank my family and friends for their never-ending encouragement and support.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

The intent of this document is to present a detailed description of the Small Scale Business Management Application. It will look to explain the purpose, functionality, design, technology used, features and desired goals of the application.
The document will also outline reflections on the development and design as it progressed and take aways during development.

1.1 Scope of Project

This software system is a Business Management Application targeted towards small scale sole trades people (e.g painter and decorator, electricians and plumbers).
The application is designed to streamline a user's business and track aspects related to the running of their business such as inventory management, job booking scheduling and planning and to allow for a more concise and easier to maintain overview of the business incomes and expenses.
The application is also intended to allow the user to keep these aspects in one easy to manage location as opposed to physical medium such as notebooks, physical calendars, receipts and quotes written down.
The system consists of a front and backend. The frontend will handle user inputs and requests from the user such as the input, update, deletion and querying of data and displaying requested and relevant information back to the user.
The frontend was developed using Bootstrap including custom CSS and Javascript file to add additional styling and functionally.
The user will not have direct access to the backend which will look after user data and will use the frontend as a mediary.
The system will allow a user to register and make a one off payment for use of the application with the implementation of a 3rd party payment process (Stripe) in order to create an account.

Once registered a user will be able to view the application Home page which will display the booked jobs which have yet to me marked as completed (currently set to display two jobs)
There will be a navbar present that will allow easy navigation.
The user will have access to a calendar component in which they can add, view, update and delete job booking from and enter information relevant to the bookings (start time, contact information, materials needed, quotes).
The user will have access to an inventory component, allowing them to keep a record of their inventory they have on hand to allow for better management of their stock and assist when planning and preparing for upcoming jobs.
The user will have access to a financial component, this will take information (quotes and material costs) from jobs booked and plot a bar graph totaling the the jobs for each month over the course of a year, allowing the user to have a clear and simplified overview of the financial aspect of the business.
The user will also be able to log out of the system with the use of the logout option in the application navbar.

The application was built using the Django which is a Python based web framework, HTML, CSS, Javascript was used for the frontend, Python and SQLite3 was used the backend to created, store, update, delete, query and created database tables along with Python also been used to manage the setting of the applications.

2 APPLICATION REQUIREMENTS

This section will outline the system requirements of the Small Scale Business Management Application and the use cases associated. The user is the main actor in the system.

2.1 System Requirements Overview

In my personal experience I have seen self employed single person businesses struggle with inventory management, job bookings and financial overviews of the business.
This system was intended to address these issues.
It would remove the need for a physical calendar which bookings are added to, amended or deleted and also capture more information in relation to a job booking than the physical calendar has space to do. The ability to edit bookings also addressed an issue of a booking being updated on a physical calendar becoming messy due to multiple edits and old information striked through or written over which lead to a cluttered and unpleasant look. With the use of the systems calendar application a user can create a booking and add the relevant info which is then saved to the calendar and displays with a booking time, the user can then select the booking and will be presented with the booking information in a clean interface, edit can then be made if necessary and any old or unneeded information can be updated or removed to provide a cleaner and more useful experience.
The inclusion of an inventory logging application was intended to assist in the planning of bookings as I have seen instances where a trades person arrives at a job to then realise they are out of a item or items that they require to complete the task at hand or when preparing for a job they would have to search through there inventory physically in order to make sure they have the materials needed. The inventory application addresses this as when planning for a booking the user can scan through the inventory system and see if they have the materials and equipment needed and their quantity and ensure they have what is required or stock up as needed before commencing the work.
The financial application is designed to give the user a clear overview of their finances per month over the year for each booking per. I have seen booking quotes written on a physical calendar along with their corresponding booking and these can fluctuate as the

work is been carried out and completed, which would lead to the original values been crossed off and new values added or the use of sticky notes been attache to the physical calendar which can come loose and then can get misplaced. Also if trying to get an overview of the income and outgoing of the business manual calculation would need to be carried out and with the somewhat haphazard way they are being logged and along with the inventory cost not being tracked in a meaningful way or at time at all it became troublesome to get a financial overview. These issues are addressed in conjunction with the calendar and booking application, quotes and materials costs can be tracked and amended as needed removing, these values are then automatically calculated and graphically depicted to give the user an easy to understand the financial incoming and outgoing expenses per month over the year.

This system is built using a Responsive Web design in mind, this approach looks to make the systems content adapt to different screen and window sizes. This will allow the system content to be viewed on a desktop and also allow the content to be viewed on smaller devices such as mobile phones or tablets, the content will then scale as needed and accommodate the device it is been viewed on without the loss of content due to elements been cut off or overlapping with each other when scaled down.

2.2 Summary Use Case Diagram

The following use case diagrams outline the functional requirements of the Small Scale Business Management Application

2.3 External Roles

The following will detail the roles and responsibilities of the use cases driven by external roles, which in this case is a user of the system.

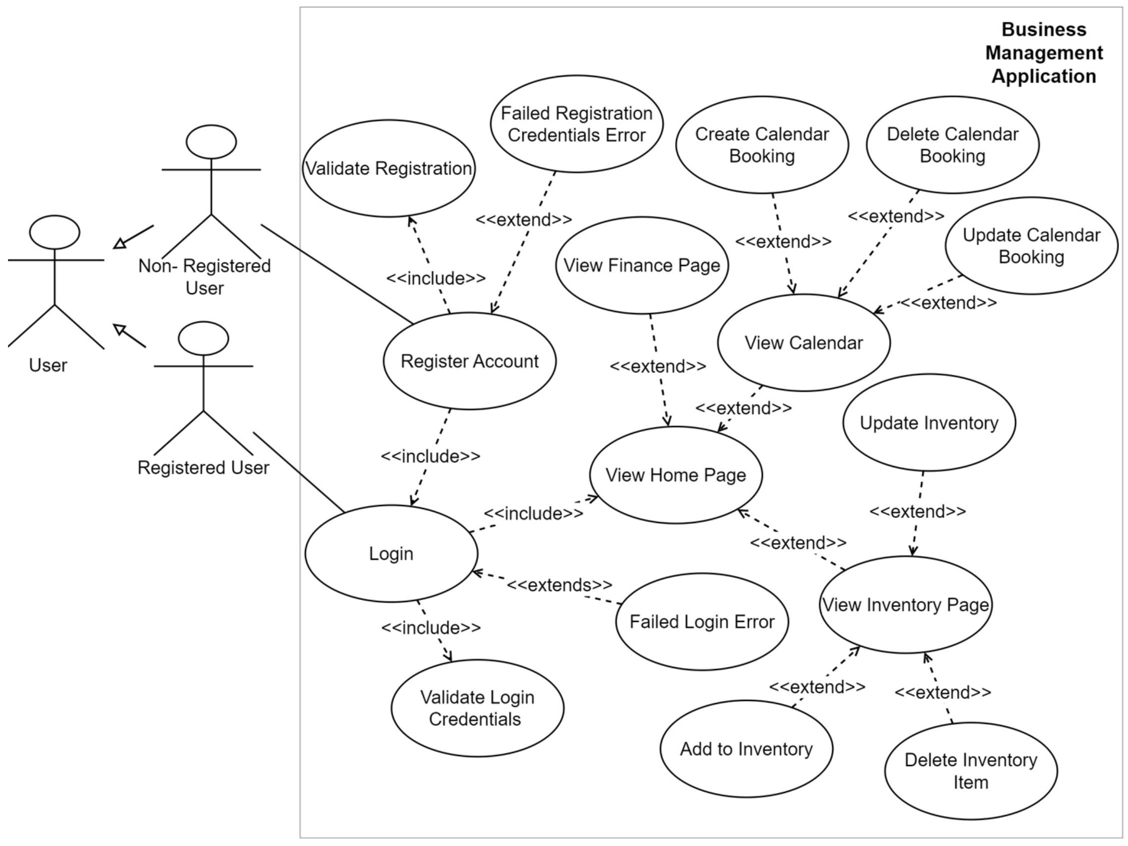


Figure 1: System Use Case Diagram

Use Cases

- User Registers an Account
- User Logins in with their Registered Account
- User Access the Home page
- User Access the Finance page
- User Access the Inventory page

- User Creates an Inventory item while on the Inventory page
- User Updates an Inventory item while on the Inventory page
- User Deletes an Inventory item while on the Inventory page
- User Access the Calendar page
- User Creates a Booking while on the Calendar page
- User Updates a Booking while on the Calendar page
- User Deletes a Booking while on the Calendar page
- User Logs out of the System

2.3 Use Case Extended Descriptions

Use Case: Register Account	
Preconditions: Unregistered users has launched the system and accessed the application Login page	
Actor: Unregistered user / Registered user	
Goals: Register a user account	
Actor Action	**System Response**
1. Unregistered user selects the “Subscribe and Register” link on the application Login page	2. User is redirected to the payment page.
3. User clicks the Checkout button	4. User is redirected to the Stripe Hosted Payment Page
5. User enters their payment information and click Pay	6. User payment is processed by Stripe (payment processing is not in the scope if the application and is handled by the integration of a 3rd party app Stripe)
	7. When payment has been processed the User is redirected to the Register page
8. User fills in the required fields on the Register page (Username, Password, Password Confirmation)	9. User input is validated and if passes validation the user is redirected to the Login Page

and selects the Complete your Registration button	
10. User enters their newly created Username and Password and click the Login button	11. User is logged in and redirected to the application Home page
Alternative Course - User account already registered	
	9. User input is validated and a pre-existing account is found for the User based on the information entered by the User (Username, Password)
	10. Username and Password is not accepted, a new account with the same credentials is not created
11. User clicks the Back button	12. User is redirected to the Login page
13. User enters their login details or can use the forgot your password link to reset their password (This feature is was not in scope)	14. User is logged in and redirected to the application Home page

Table 1: Register Account

Use Case: Login	
Preconditions: Registered users has launched the system and accessed the application Login page	
Actor: Registered user	
Goals: Login to application	
Actor Action	**System Response**
1. Users enters Username and Password	2. User login input is validated and the User is redirected the the application home page
Alternative Course - User Fails Authentication	
	2. User login inputs fails validation

	3. User is advised there is an issue with their login details
4. User reattempts login or uses the forgot your password link to reset their password (This feature is was not in scope)	
5. User uses their correct valid credentials	6. User is logged in and redirected the the application home page

Table 2: Login

Use Case: View Calendar	
Preconditions: User has launched the system and logged in and is on the application Home page	
Actor: Registered user	
Goals: View Calendar Page	
Actor Action	**System Response**
1. Users selects Calendar from the Home Page Navbar	2. User is redirected the the Calendar page

Table 3: View Calendar Page

Use Case: Create Calendar Booking	
Preconditions: User has launched the system and logged in and navigated to the Calendar page	
Actor: Registered user	
Goals: Create Calendar Booking	
Actor Action	**System Response**
1. Users clicks the New Booking button	2. User is redirected the to Create/Edit booking page
3. Users enters details of a booking in the required fields displayed (Title, Details, Contact, Phone number, Quote, Materials needed, Materials	4. Details for the booking are saved to the Calendar database, User is redirected the the Calendar page and the new booking is displayed on the Calendar

cost, Start time, End time) and clicks the Save button	
Alternative Course - User attempts to save a Booking without filling in required fields	
3. Users enters no details for a booking in one or more of the required fields (Title, Details, Contact, Phone number, Quote, Materials needed, Materials cost, Start time, End time) and clicks the Save button	4. A warning message is displayed alerting the User to the fields that require details, no updates are saved to the Calendar database

Table 4: Create Calendar Booking

Use Case: Update a Calendar Booking	
Preconditions: User has launched the system and logged in and navigated to the Calendar page	
Actor: Registered user	
Goals: Edit Calendar Booking	
Actor Action	**System Response**
5. Users clicks an existing Booking on the Calendar	6. User is redirected the to Create/Edit booking page, Booking details are populated
7. Users edits details of the selected booking and clicks the Save button	8. Details for the edited booking are saved to the Calendar database, User is redirected the the calendar page and the edited booking is displayed on the Calendar

Table 5: Update a Calendar Booking

Use Case: Delete Calendar Booking	
Preconditions: User has launched the system and logged in and navigated to the Calendar page	
Actor: Registered user	
Goals: Delete Calendar Booking	
Actor Action	**System Response**

9. Users clicks an existing Booking on the Calendar	10. User is redirected the to Create/Edit booking page, Booking details are populated
11. Users clicks the Delete button	12. A confirm delete modal is displayed
13. User clicks the Confirm button	14. Modal closes and Booking is deleted from the Calendar database and the User is redirected to the Calendar page, deleted booking is no longer displayed on the Calendar
Alternative Course - User does not confirm deletion of Booking	
13. Users clicks the Cancel Button	14. confirm delete modal closes and User can then navigate to the Calendar page by clicking the Back button on the Create/Edit booking page
	15. Booking remind on the Calendar page

Table 6: Delete Calendar Booking

Use Case: View Inventory Page	
Preconditions: User has launched the system and logged in and is on the application Home page	
Actor: Registered user	
Goals: View Inventory	
Actor Action	**System Response**
1. Users selects Inventory from the Home Page Navbar	2. User is redirected the the Inventory page

Table 7: View Inventory Page

Use Case: Add a new Item to Inventory	
Preconditions: User has launched the system and logged in and navigated to the Inventory page	
Actor: Registered user	
Goals: Add a new Item to Inventory	
Actor Action	**System Response**
1. User clicks the Add New Item button	2. User is redirected to the Add new Inventory Item page
3. User enters details for the Inventory item in the required fields (Item Name, Item Description, Item Quantity) and clicks the Save button	4. New Inventory Item is saved to the Inventory database and the user is redirected to the Inventory page and the new Inventory item has been added.
Alternative Course - User attempts to save a New Inventory item without filling in required fields	
3. User does not enter Inventory Item details for one or more required fields and clicks the Save button.	4. A warning message is displayed alerting the User to the fields that require details

Table 8: Add a new Item to Inventory

Use Case: Update an Inventory item	
Preconditions: User has launched the system and logged in and navigated to the Inventory page	
Actor: Registered user	
Goals: Update an Inventory item	
Actor Action	**System Response**
1. User selected the Update button associated to the Inventory item they wish to update	2. User is redirected to the edit Inventory Item page, the selected Inventory item details are populated
3. Users edits details of the selected Inventory item and clicks the Save button	4. Details for the edited Inventory item are saved to the Inventory database, User is redirected the the Inventory page and the edited

	Inventory item is displayed
Alternative Course - User attempts to save a New Inventory item without filling in required fields	
3. User does not enter Inventory Item details for one or more required fields and clicks the Save button.	4. A warning message is displayed alerting the User to the fields that require details, no updates for the Inventory item are saved to the Inventory database

Table 9: Update an Inventory item

Use Case: Delete an Inventory item	
Preconditions: User has launched the system and logged in and navigated to the Inventory page	
Actor: Registered user	
Goals: Delete an Inventory item	
Actor Action	**System Response**
1. User selected the Delete button associated to the Inventory item they wish to delete	2. A confirm delete modal is displayed
3. User clicks the Confirm button	4. Modal closes and the Inventory item is deleted from the Inventory database and the Inventory item deleted from the Inventory page
Alternative Course - User does not confirm deletion of Inventory Item	
3. Users clicks the Cancel Button	4. Modal closes, Inventory item is not deleted from the Inventory database

Table 10: Delete an Inventory item

Use Case: View Finance Page	
Preconditions: User has launched the system and logged in and is on the application Home page	
Actor: Registered user	
Goals: View Finance	
Actor Action	**System Response**
1. User selects Finance from the Home page Navbar	2. User is redirected to the Finance page

Table 11: View Finance Page

Use Case: Logout of Application	
Preconditions: User has launched the system and logged in and is on the application Home page	
Actor: Registered user	
Goals: Logout of application	
Actor Action	**System Response**
1. Users selects Logout from the Home page Navbar	2. User is logged out of the application

Table 12: Logout of Application

2.4 User Interface

Each page is intended to be user friendly, uncluttered and easy to navigate to and from. The following are mockups for the pages a user can access. All the mockups were created using diagrams.net. Diagrams.net is a web based application that allows for the creation and editing of various types of diagrams such as wireframe and UML diagrams.

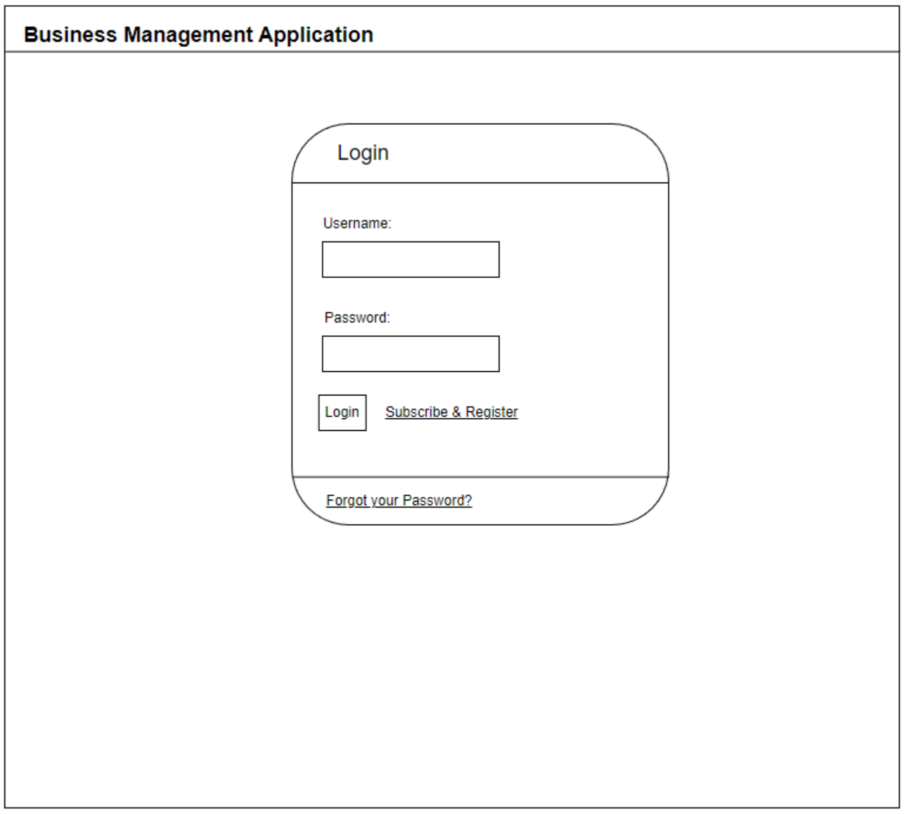


Figure 2: Login Page

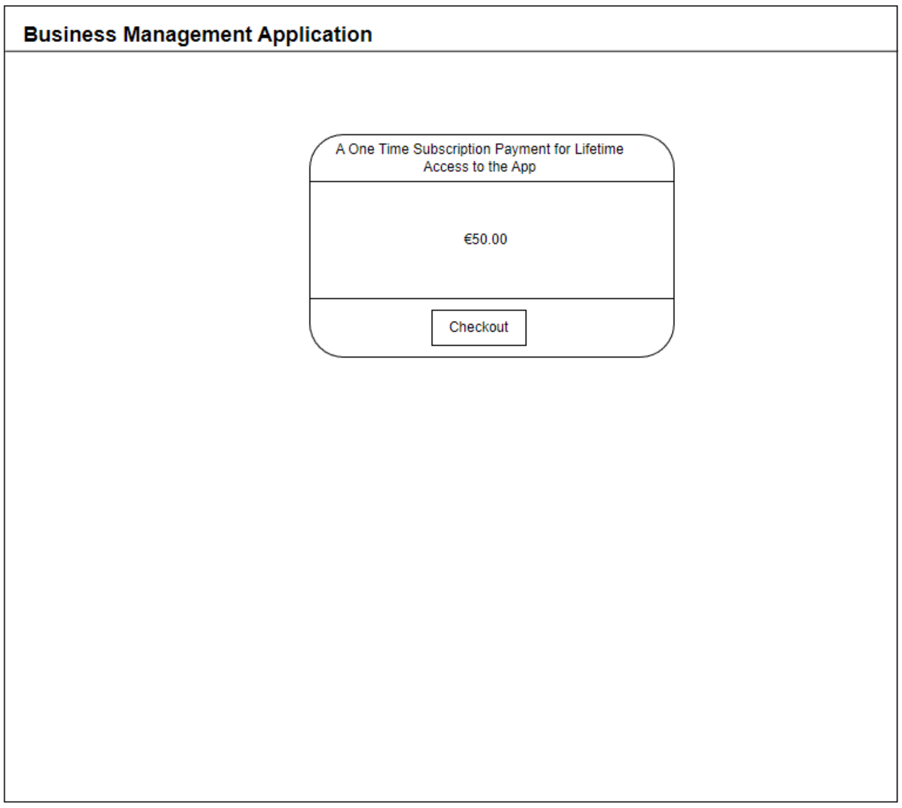

Business Management Application
A One Time Subscription Payment for Lifetime Access to the App
€50.00
Checkout

Figure 3: Subscription Page

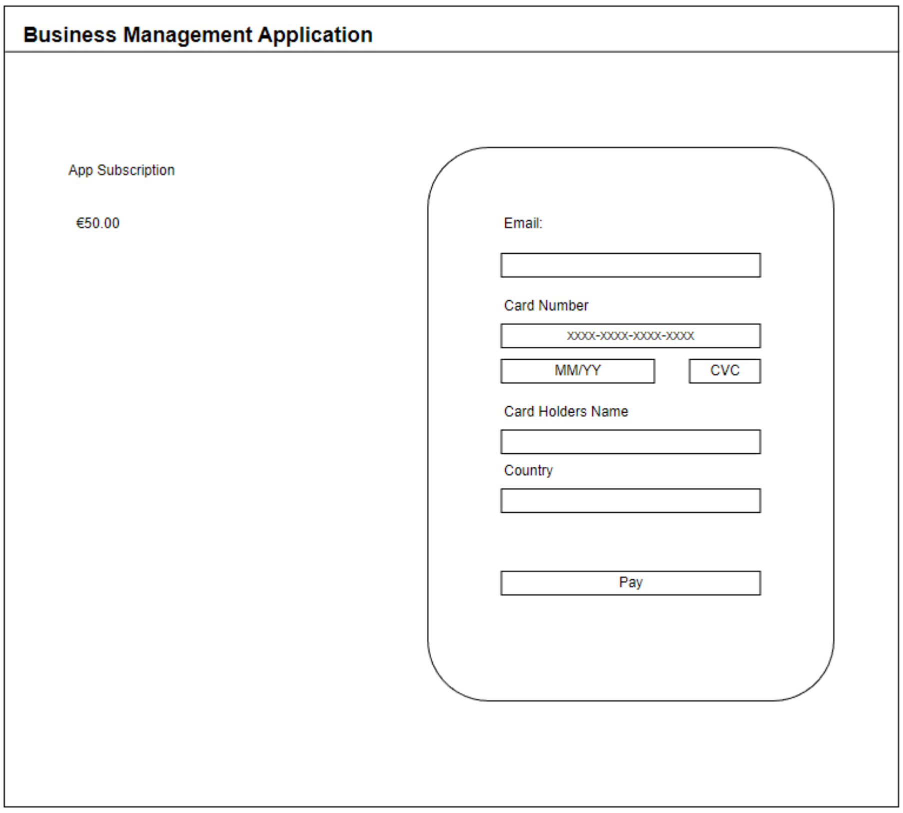


Figure 4: Payment Page

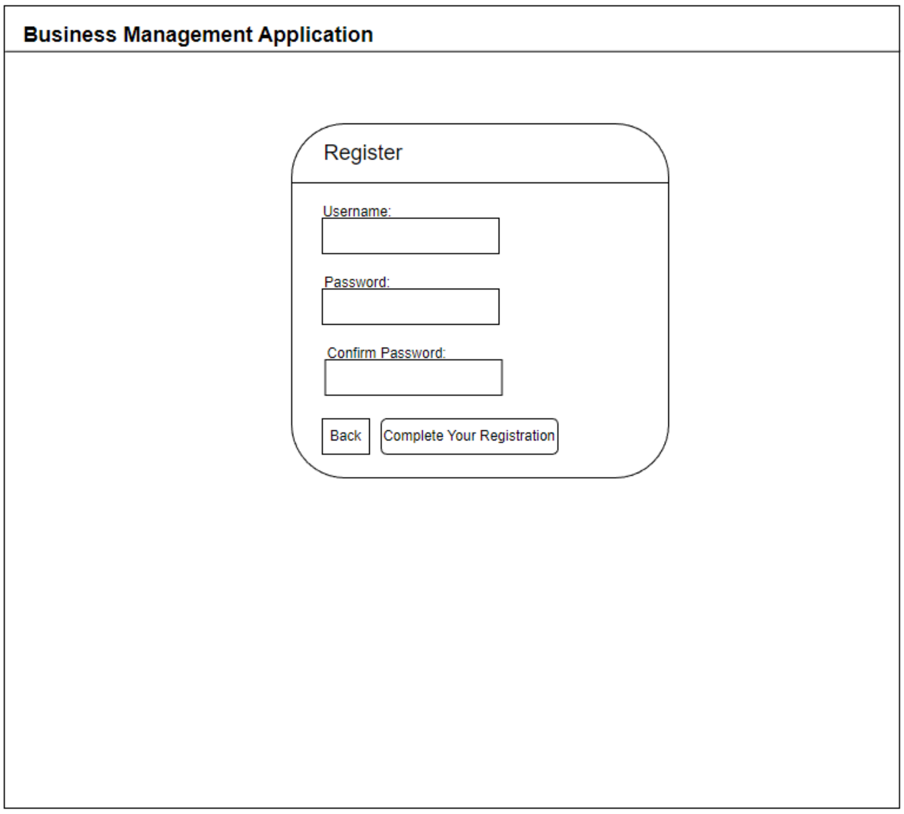
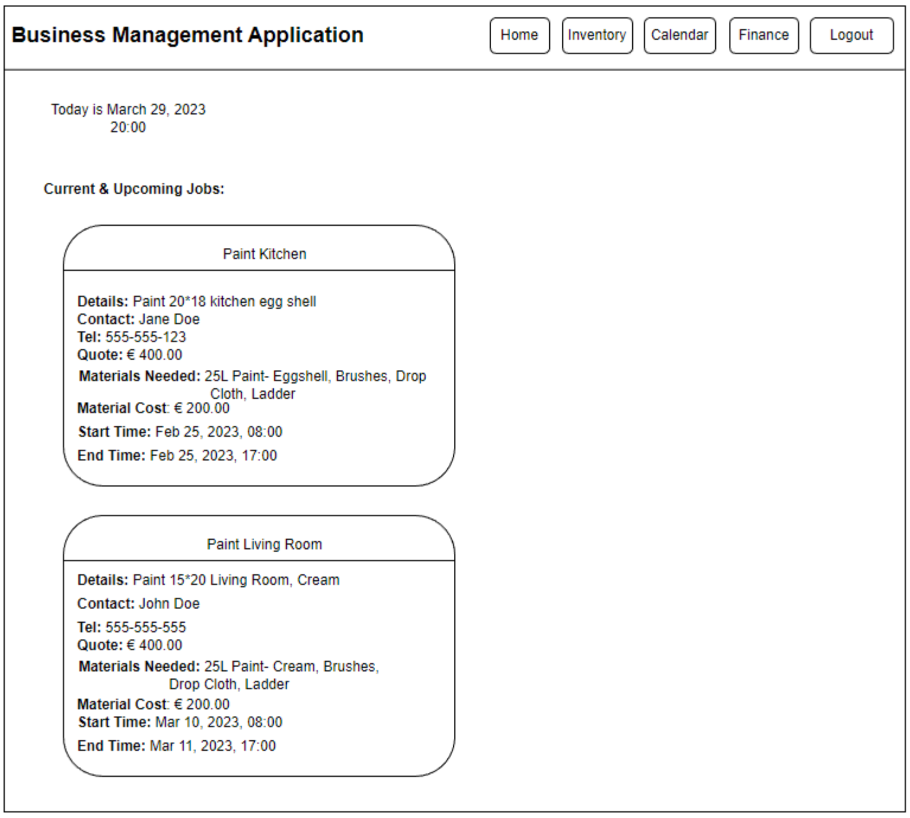


Figure 5: Registration Page



Today is March 29, 2023
20:00

Current & Upcoming Jobs:

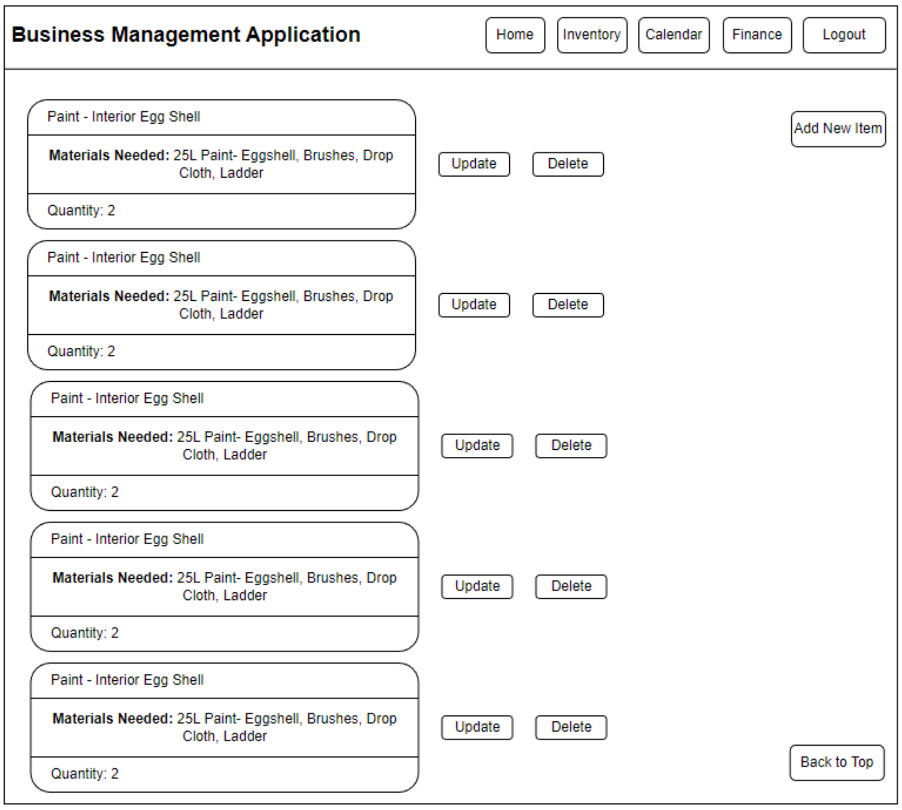
Paint Kitchen

Details: Paint 20*18 kitchen egg shell
Contact: Jane Doe
Tel: 555-555-123
Quote: € 400.00
Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder
Material Cost: € 200.00
Start Time: Feb 25, 2023, 08:00
End Time: Feb 25, 2023, 17:00

Paint Living Room

Details: Paint 15*20 Living Room, Cream
Contact: John Doe
Tel: 555-555-555
Quote: € 400.00
Materials Needed: 25L Paint- Cream, Brushes, Drop Cloth, Ladder
Material Cost: € 200.00
Start Time: Mar 10, 2023, 08:00
End Time: Mar 11, 2023, 17:00

Figure 6: Home Page



Paint - Interior Egg Shell

Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder

Quantity: 2

Paint - Interior Egg Shell

Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder

Quantity: 2

Paint - Interior Egg Shell

Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder

Quantity: 2

Paint - Interior Egg Shell

Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder

Quantity: 2

Paint - Interior Egg Shell

Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder

Quantity: 2

Figure 7: Inventory Page

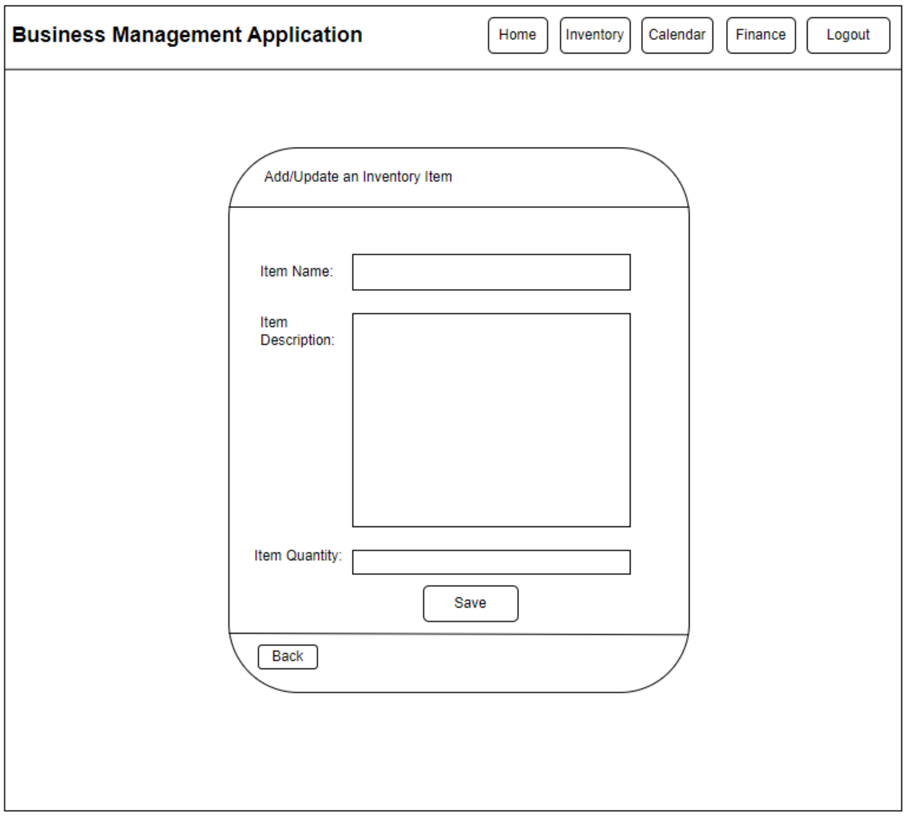


Figure 8: Add Inventory Item Page

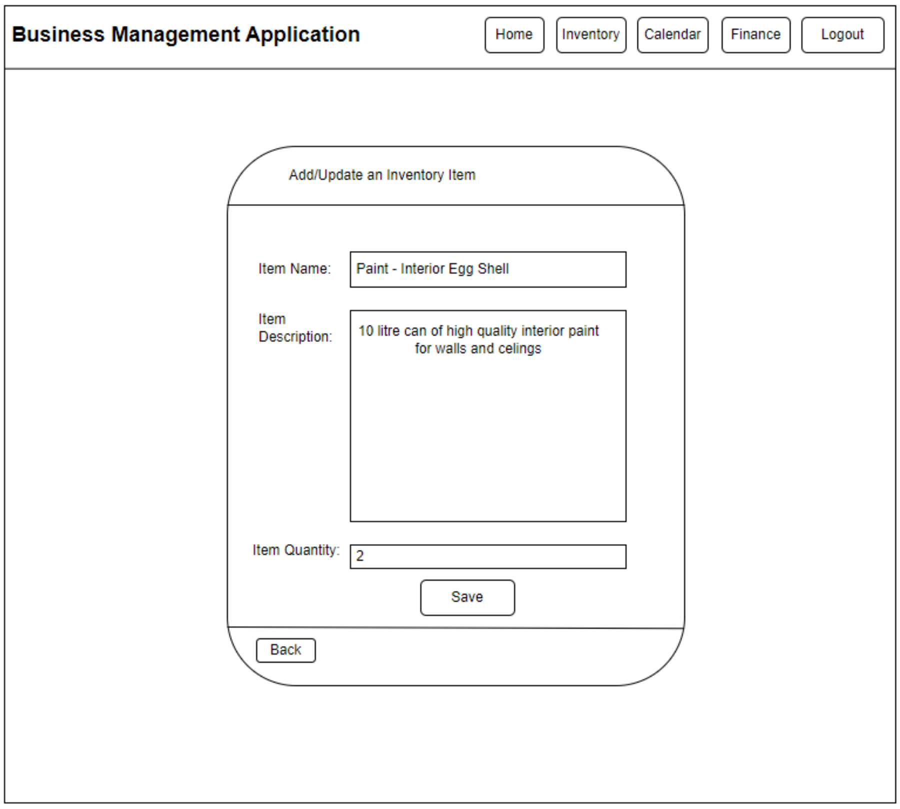

Business Management Application
Home
Inventory
Calendar
Finance
Logout
Add/Update an Inventory Item
Item Name:
Paint - Interior Egg Shell
Item Description:
10 litre can of high quality interior paint for walls and celings
Item Quantity:
2
Save
Back

Figure 9: Edit Inventory Item Page

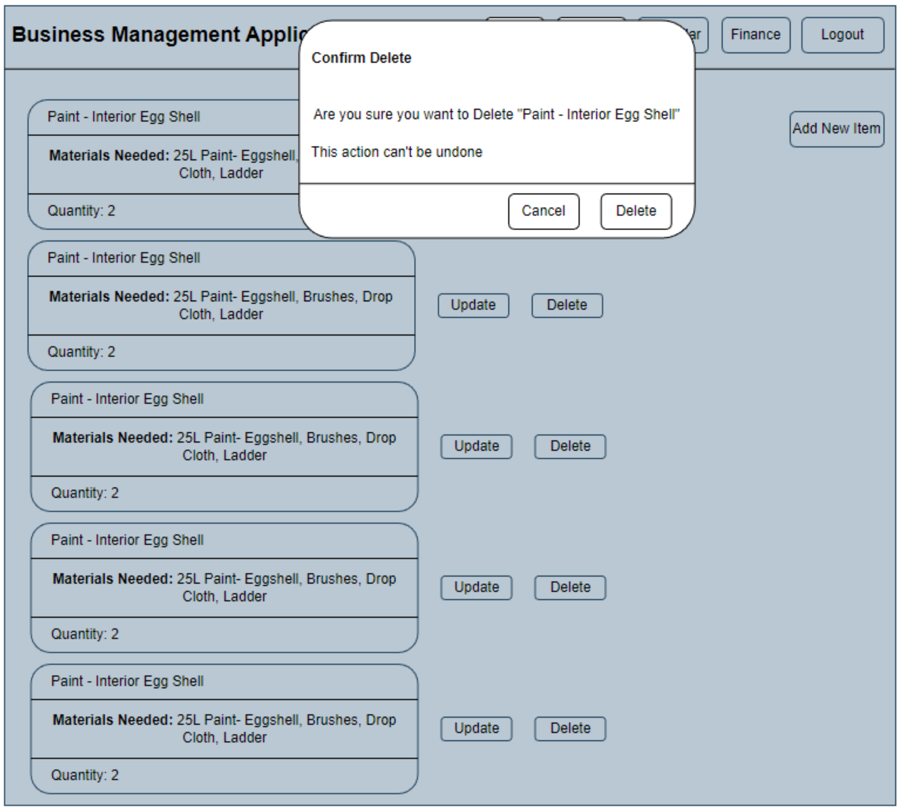

Business Management Applic
Finance
Logout
Confirm Delete
Are you sure you want to Delete "Paint - Interior Egg Shell"
This action can't be undone
Cancel
Delete
Add New Item
Paint - Interior Egg Shell
Materials Needed: 25L Paint- Eggshell, Brushes, Drop Cloth, Ladder
Quantity: 2
Update
Delete

Figure 10: Inventory Item Delete Confirmation Modal

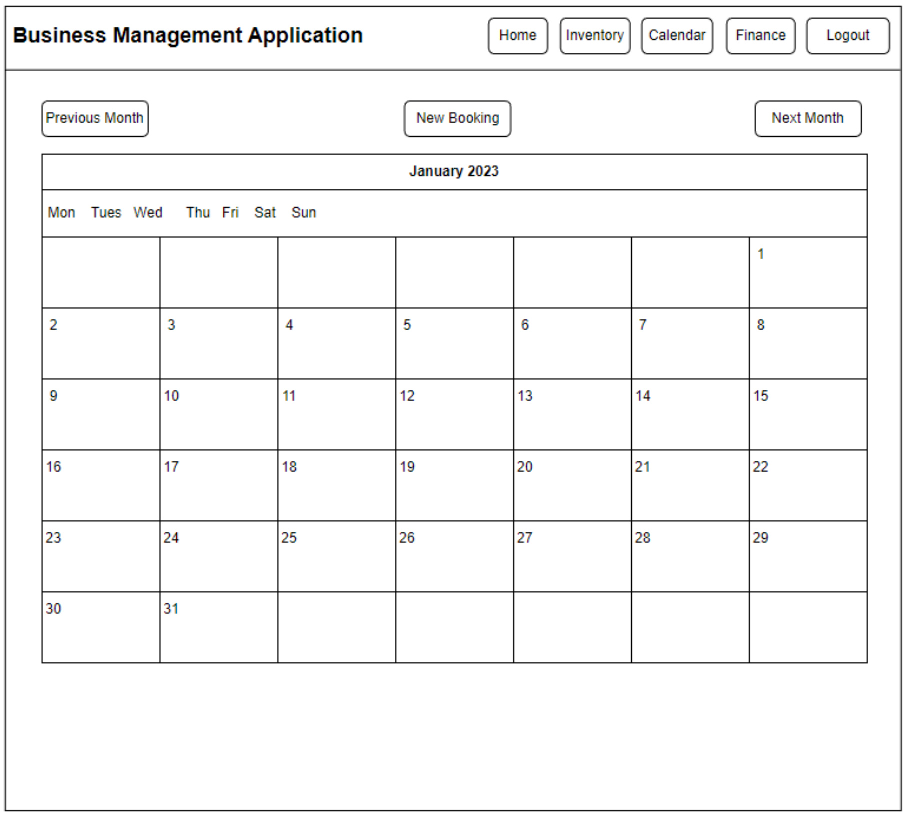


Figure 11: Calendar Page

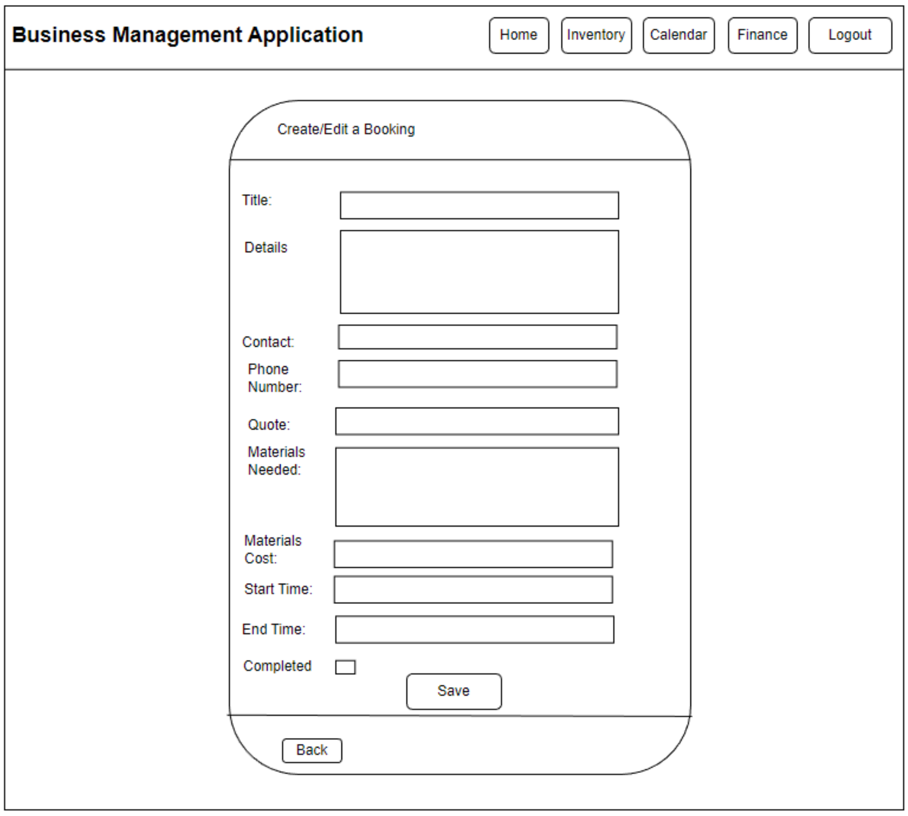
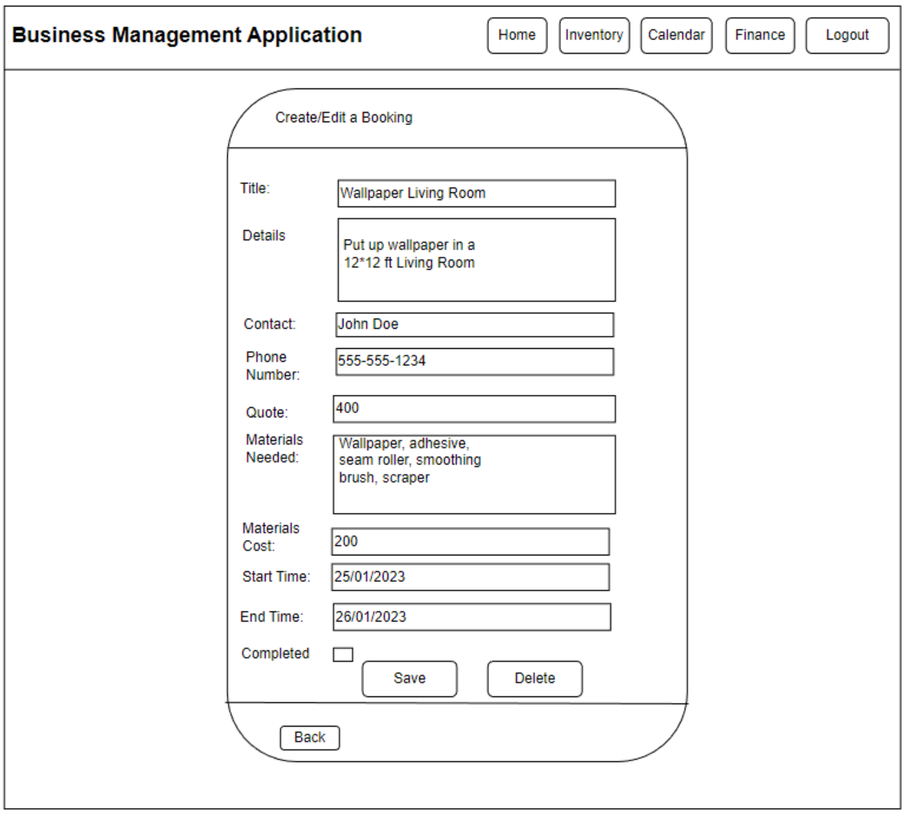


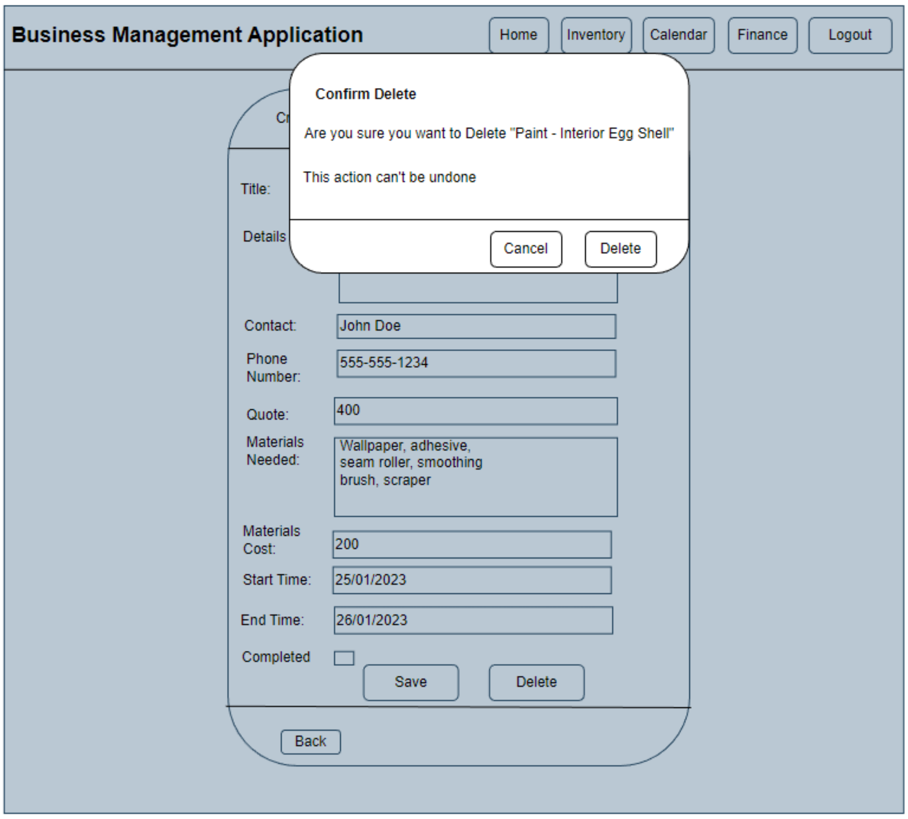
Figure 12: Add Calendar Booking Page


Business Management Application

Create/Edit a Booking

Title:	Wallpaper Living Room
Details	Put up wallpaper in a 12*12 ft Living Room
Contact:	John Doe
Phone Number:	555-555-1234
Quote:	400
Materials Needed:	Wallpaper, adhesive, seam roller, smoothing brush, scraper
Materials Cost:	200
Start Time:	25/01/2023
End Time:	26/01/2023
Completed	☐

Figure 13: Edit Calendar Booking Page


Business Management Application

Confirm Delete

Are you sure you want to Delete "Paint - Interior Egg Shell"

This action can't be undone

Title:	
Details	
Contact:	John Doe
Phone Number:	555-555-1234
Quote:	400
Materials Needed:	Wallpaper, adhesive, seam roller, smoothing brush, scraper
Materials Cost:	200
Start Time:	25/01/2023
End Time:	26/01/2023
Completed	☐

Figure 14: Calendar Booking Delete Confirmation Modal

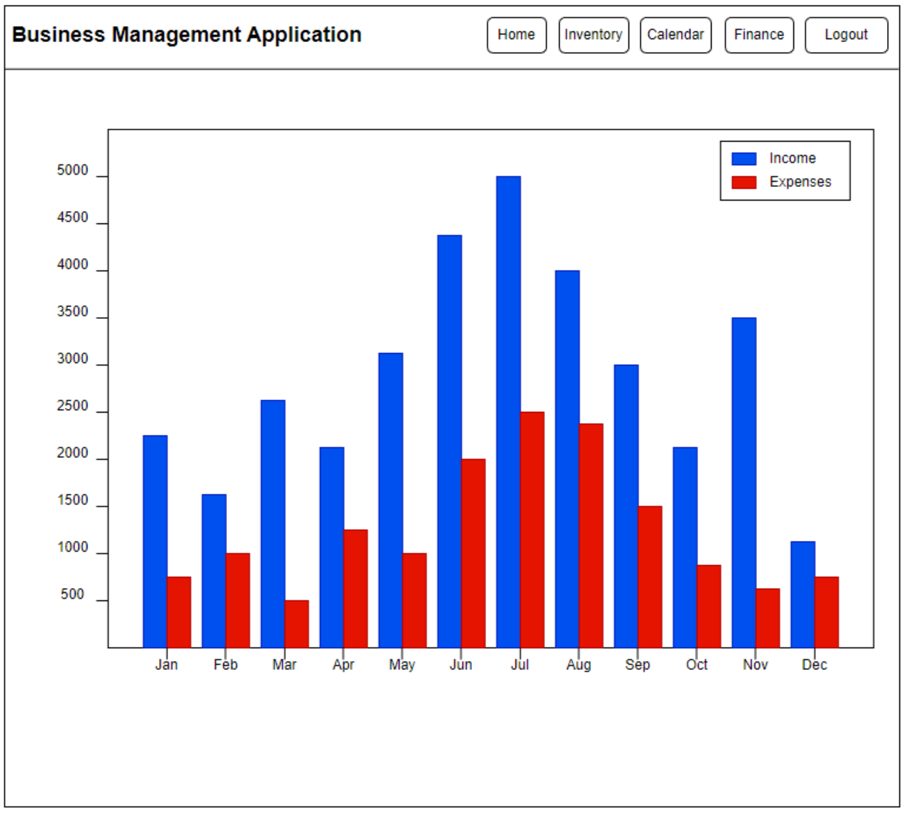


Figure 15: Finance Page

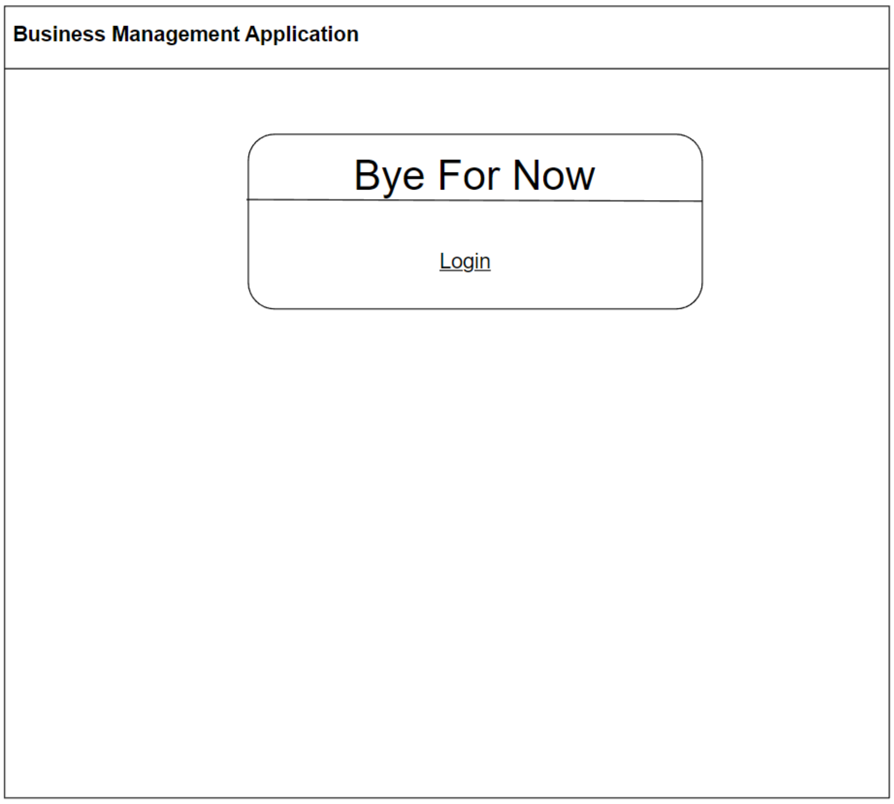


Figure 16: Logout Page

2.5 Non Functional Requirements

The application should be fully operational and accessible once activated. Pages should load almost instantaneously when navigated to. The application should be reliable and function without error and alert the user when an issue is encountered. It should also offer portability due the responsive design implemented. The application should also be able to manage multiple users at a time and request from different users simultaneously. Also due to the use of Django and the Model View Template architecture used in the applications development it should allow scalability should the need arise.

2.6 Safety and Security Requirements

Security requirements implemented in the system are as follows:

- A user is restricted from full access to the system prior to registering an account. When a user is registered and signed in will have access to the Home page, Inventory Page, Calendar page and Finance page. If an unregistered user or a registered user that is not logged in attempts to access these pages via their URLs they will be redirected to the applications Login page where they can then register an account or log if they are already registered.
- A registered user's information is not accessible or viable to another registered user. If user A addes items to their inventory or creates job bookings, this will not be visible to user B, user B will only see their own information and data added to their account.
- The use of CSRF tokens in form submission. CSRF (Cross Site Request Forgery) tokens are a security feature used to protect web applications from Cross Site Request Forgery attacks in which an attacker attempts to trick a registered user into executing a malicious action on the application without the user's knowledge. The use of a unique token attached to form submissions can help with this as when a form is submitted Django will look for this unique token and if not present treats the form submission as malicious and rejects it.

3 DESIGN

This application was developed using the Django web framework.

The following is the class diagram showing the associations and relationships between classes

3.1 Class Diagram

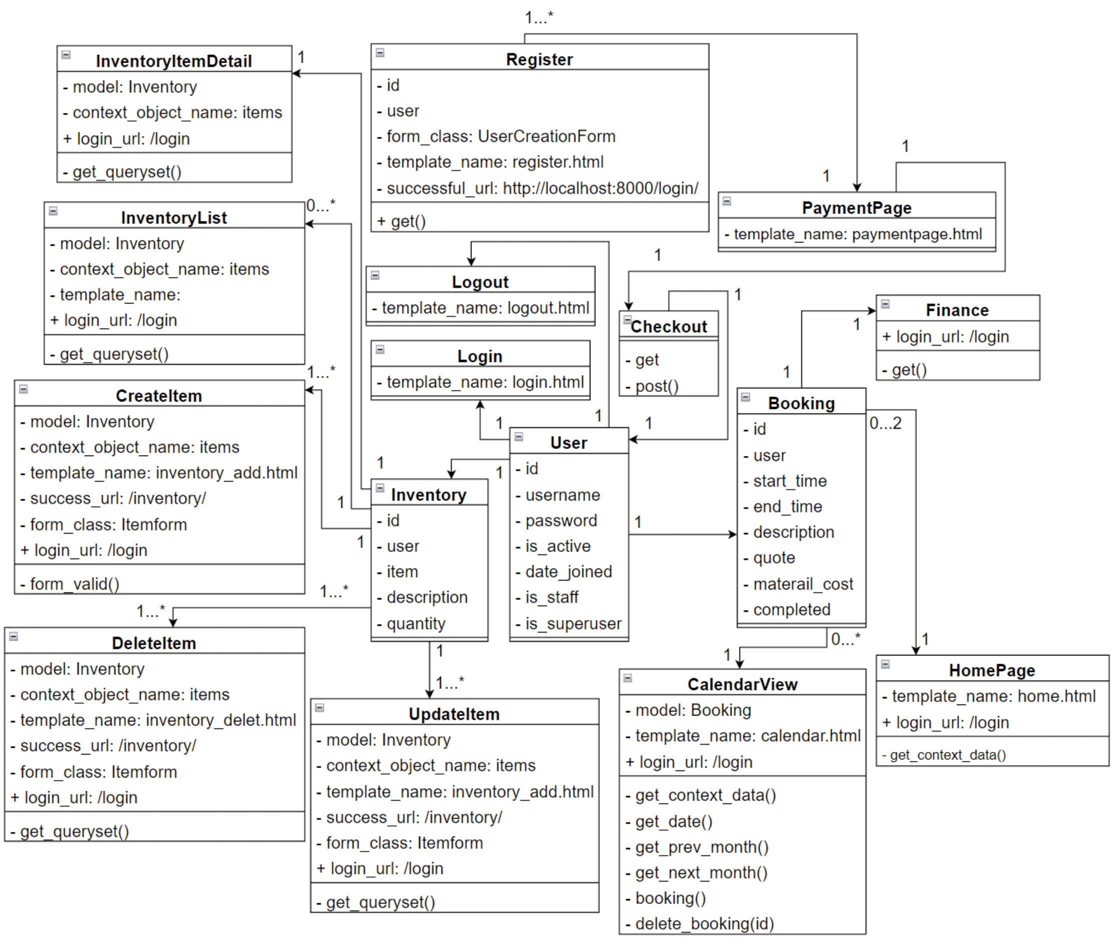


Figure 17 : Small Scale Business Management Application Class Diagram

3.2 State Diagram

The following state diagrams illustrate states that can be moved through by the Calendar booking, Inventory management and User Login.

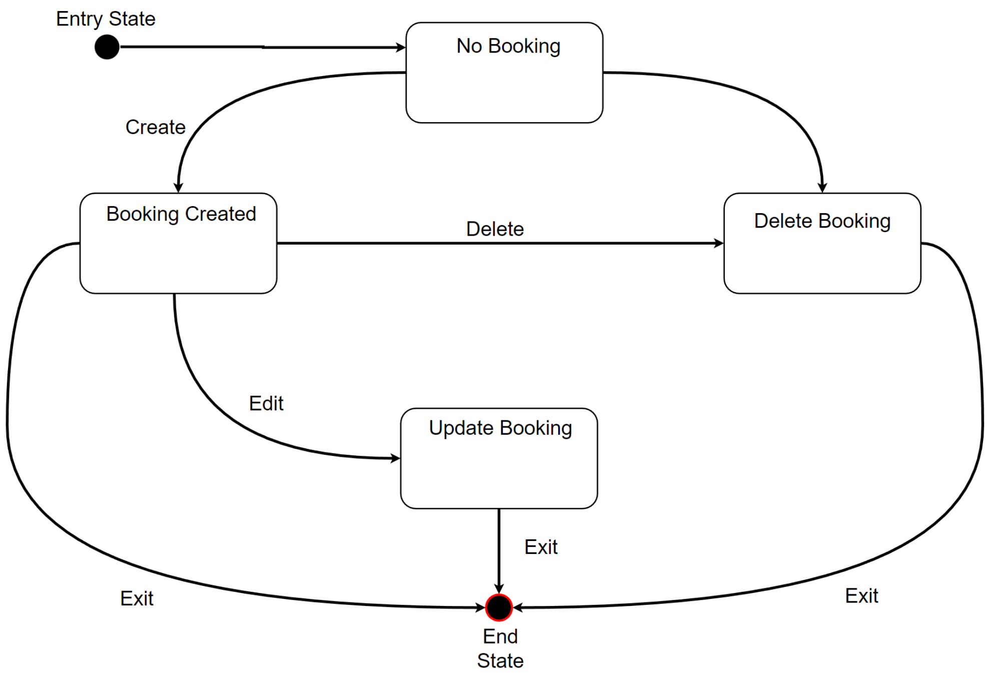


Figure 18 : Calendar State Diagram

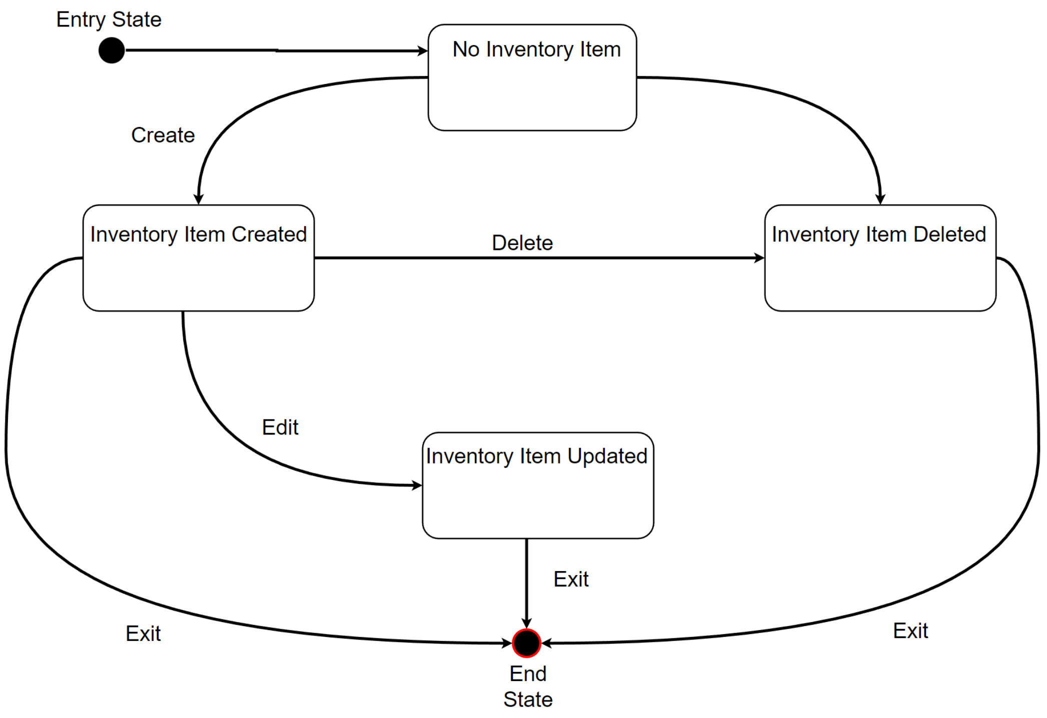
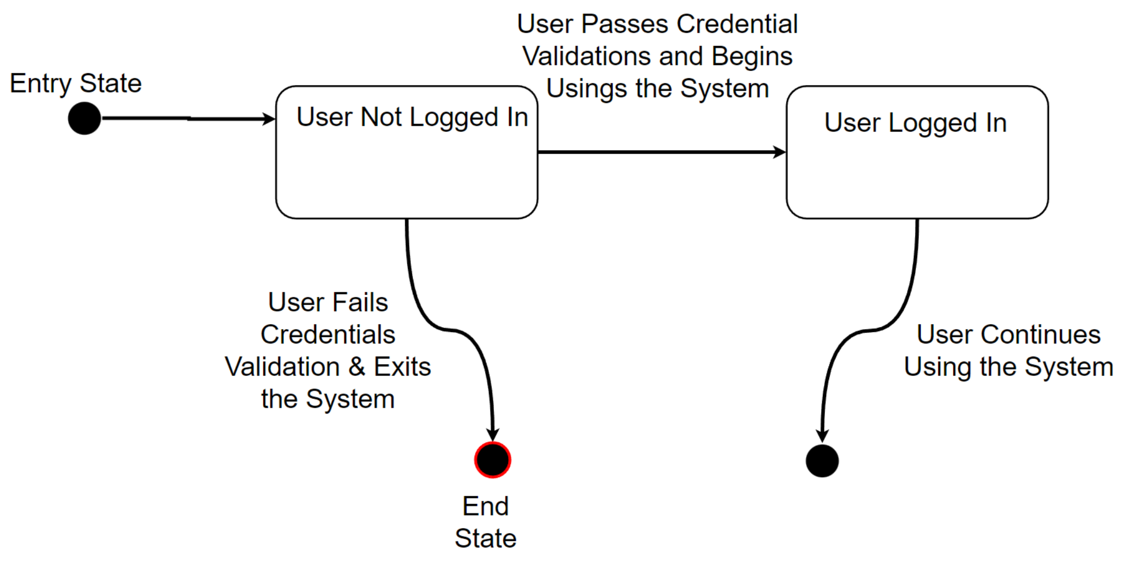


Figure 19 : Inventory State Diagram



Entry State
User Not Logged In
User Passes Credential Validations and Begins Usings the System
User Logged In
User Fails Credentials Validation & Exits the System
User Continues Using the System
End State

Figure 20 : Login State Diagram

3.3 Sequence Diagram

The following sequence diagrams illustrate the various sequence of events a user and the system undertake when in use

3.3.1 View Inventory Items

The View Inventory Items sequence diagram illustrates a user viewing the Inventory page

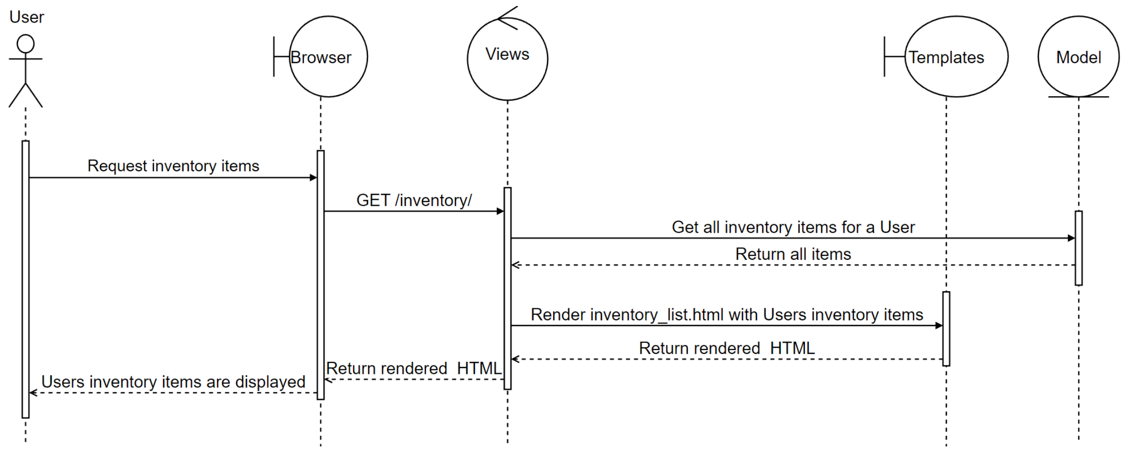


Figure 21 : View Inventory Items Sequence Diagram

The Update Inventory Items sequence diagram illustrates a user updating an items on the Inventory page

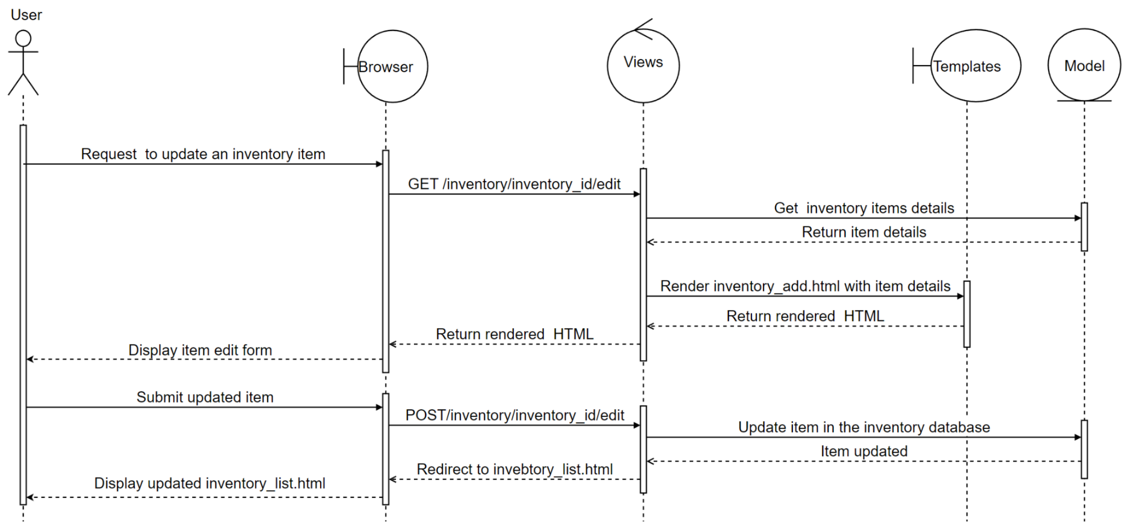

User
Browser
Views
Templates
Model
Request to update an inventory item
GET /inventory/inventory_id/edit
Get inventory items details
Return item details
Render inventory_add.html with item details
Return rendered HTML
Return rendered HTML
Display item edit form
Submit updated item
POST/inventory/inventory_id/edit
Update item in the inventory database
Item updated
Redirect to invebtory_list.html
Display updated inventory_list.html

Figure 22 : Update Inventory Items Sequence Diagram

The Delete Inventory Items sequence diagram illustrates a user deleting an items on the Inventory page

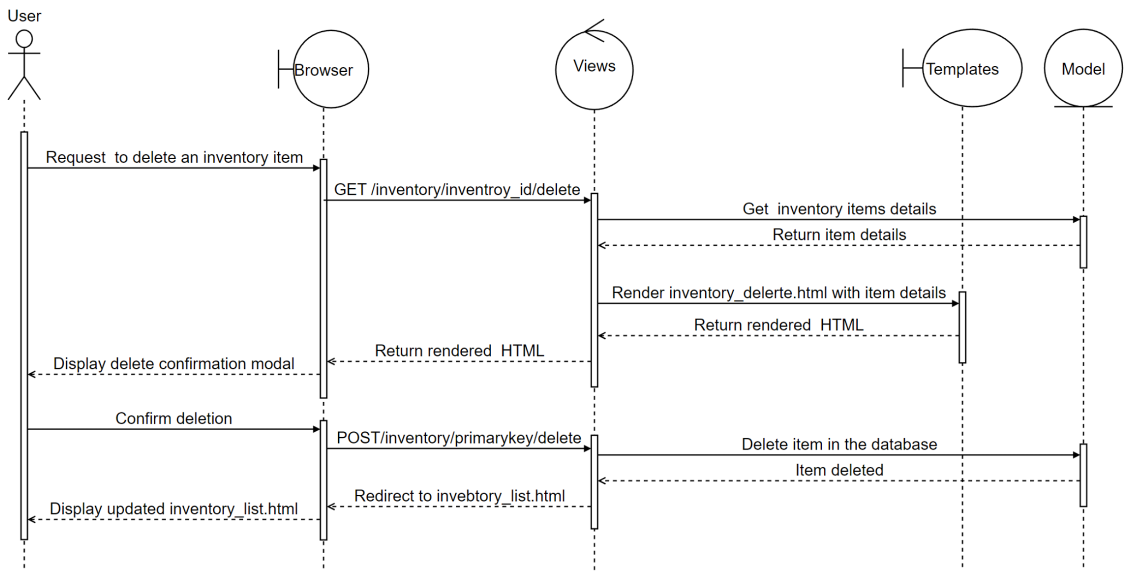

User
Browser
Views
Templates
Model
Request to delete an inventory item
GET /inventory/inventroy_id/delete
Get inventory items details
Return item details
Render inventory_delerte.html with item details
Return rendered HTML
Return rendered HTML
Display delete confirmation modal
Confirm deletion
POST/inventory/primarykey/delete
Delete item in the database
Item deleted
Redirect to invebtory_list.html
Display updated inventory_list.html

Figure 23 : Delete Inventory Items Sequence Diagram

The Add Inventory Items sequence diagram illustrates a user adding an items on the Inventory page

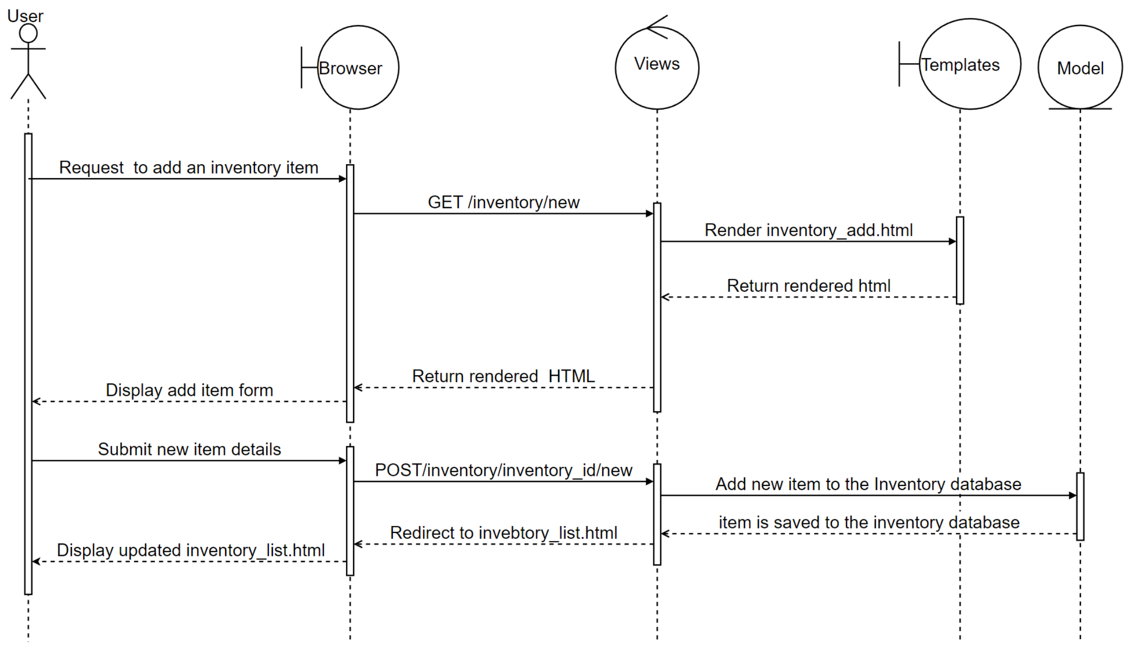

User
Browser
Views
Templates
Model
Request to add an inventory item
GET /inventory/new
Render inventory_add.html
Return rendered html
Return rendered HTML
Display add item form
Submit new item details
POST/inventory/inventory_id/new
Add new item to the Inventory database
item is saved to the inventory database
Redirect to invebtory_list.html
Display updated inventory_list.html

Figure 24 : Adding Inventory Items Sequence Diagram

The View Calendar sequence diagram illustrates a user viewing the Calendar page

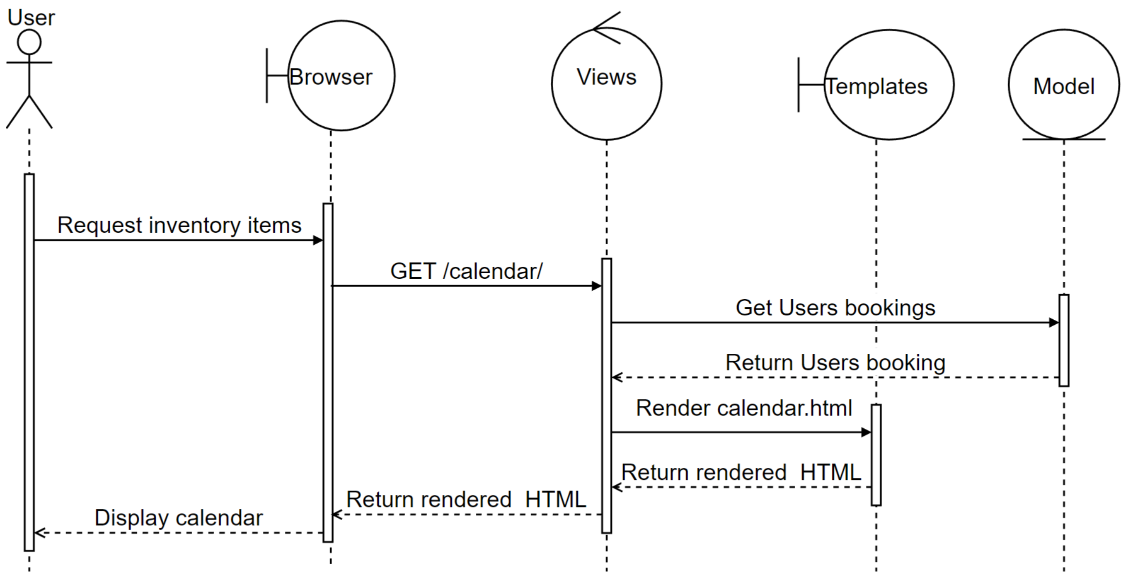


Figure 25 : View Calendar Sequence Diagram

The Update Calendar booking sequence diagram illustrates a user updating an existing booking on the Calendar page

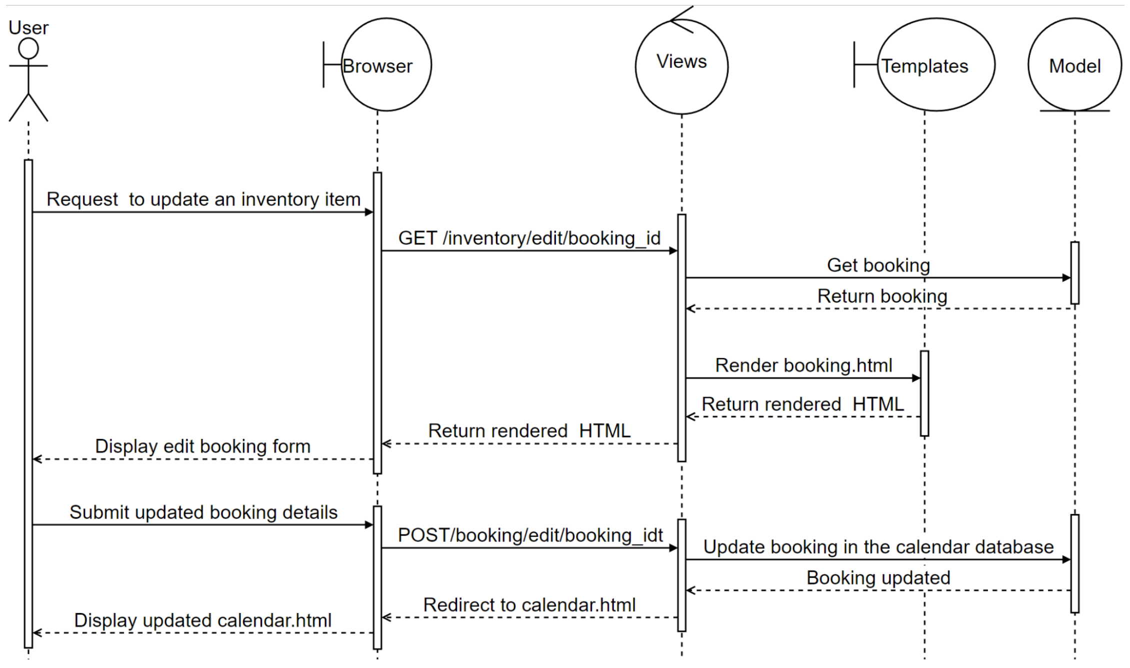

User
Browser
Views
Templates
Model
Request to update an inventory item
GET /inventory/edit/booking_id
Get booking
Return booking
Render booking.html
Return rendered HTML
Return rendered HTML
Display edit booking form
Submit updated booking details
POST/booking/edit/booking_idt
Update booking in the calendar database
Booking updated
Redirect to calendar.html
Display updated calendar.html

Figure 26 : Update a Calendar Booking Sequence Diagram

The Delete Calendar booking sequence diagram illustrates a user deleting an existing booking on the Calendar page

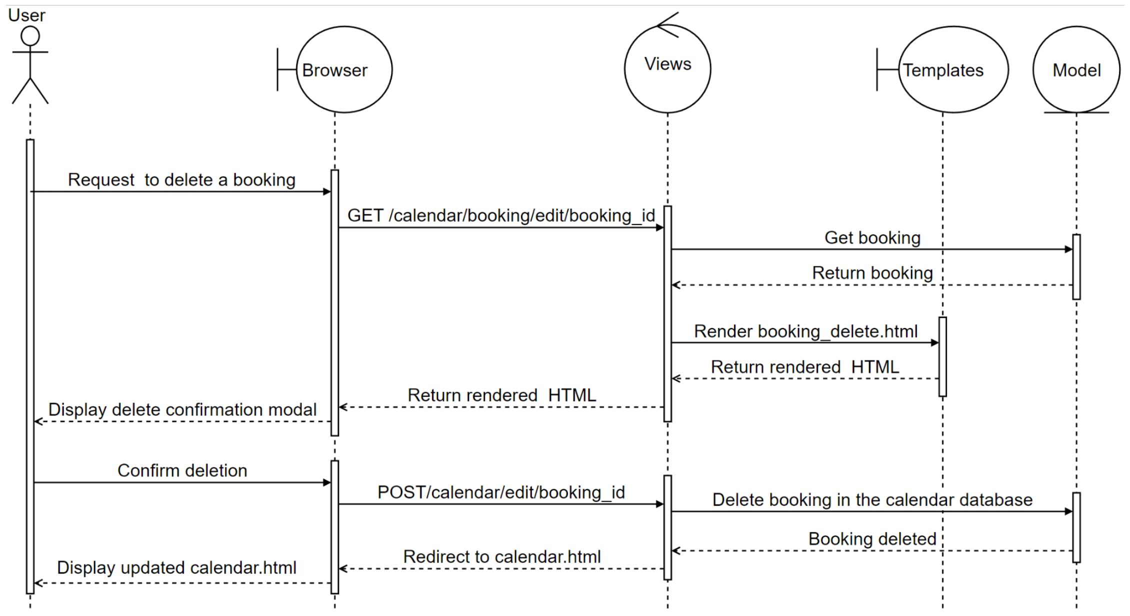
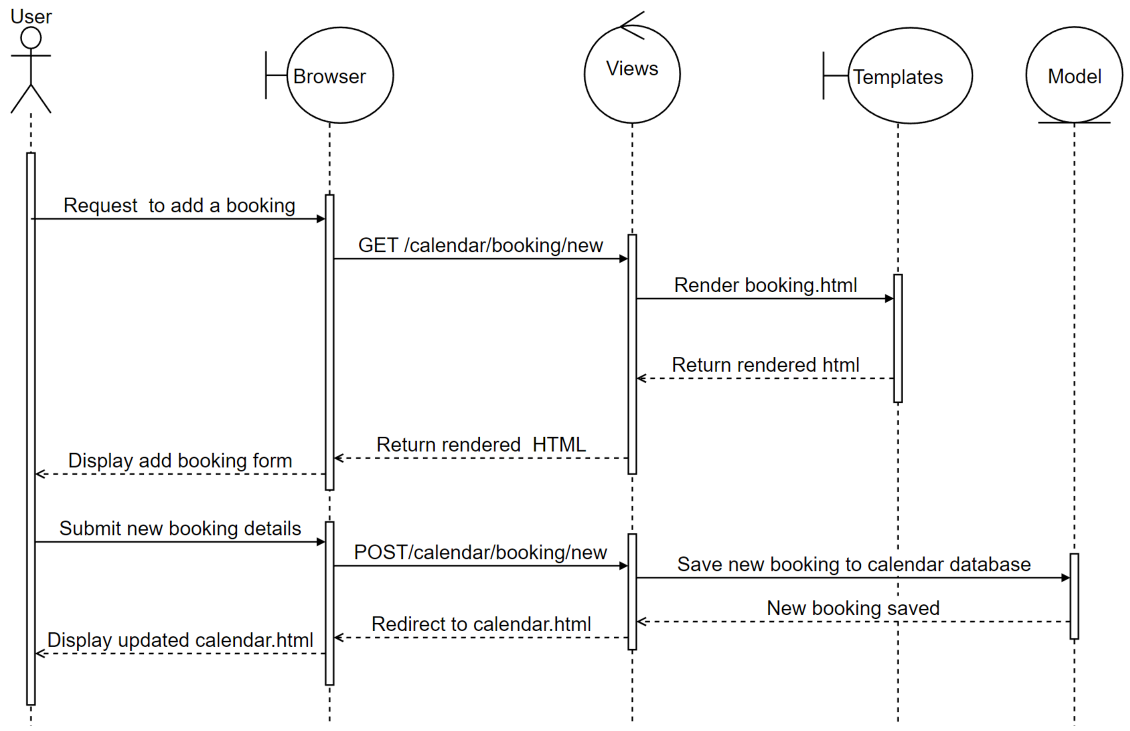

User
Browser
Views
Templates
Model
Request to delete a booking
GET /calendar/booking/edit/booking_id
Get booking
Return booking
Render booking_delete.html
Return rendered HTML
Return rendered HTML
Display delete confirmation modal
Confirm deletion
POST/calendar/edit/booking_id
Delete booking in the calendar database
Booking deleted
Redirect to calendar.html
Display updated calendar.html

Figure 27 : Delete a Calendar Booking Sequence Diagram

The Add Calendar booking sequence diagram illustrates a user Adding a booking to the Calendar page



User
Browser
Views
Templates
Model
Request to add a booking
GET /calendar/booking/new
Render booking.html
Return rendered html
Return rendered HTML
Display add booking form
Submit new booking details
POST/calendar/booking/new
Save new booking to calendar database
New booking saved
Redirect to calendar.html
Display updated calendar.html

Figure 28 : Add a Calendar Booking Sequence Diagram

3.4 Entity Relation Diagram

Database Design

The application database SQLite3 which is a open-source lightweight serverless database that is preconfigured as the default for Django.

The requirements for the databases were outlined during the planning stages, these requirements were then created using Python class and removed the need to write SQL code in the development of the database and could also be enhanced with ease with the import of Python modules which again minimised the design and implementation effort of the database.

The database allows for create, read, update and delete functionality.

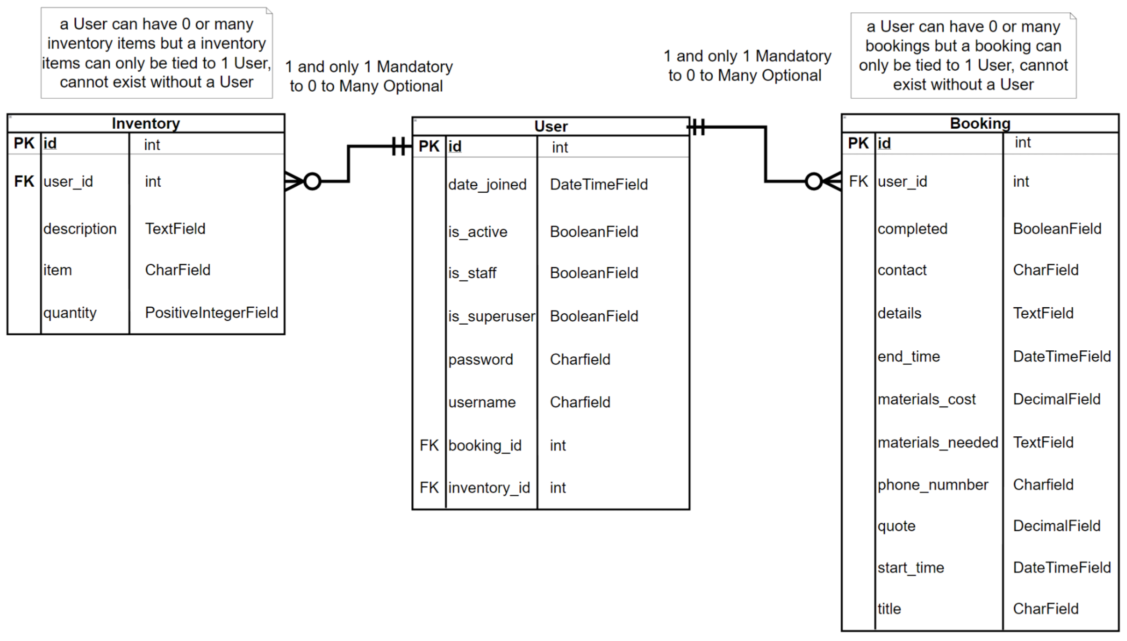

a User can have 0 or many inventory items but a inventory items can only be tied to 1 User, cannot exist without a User
1 and only 1 Mandatory to 0 to Many Optional
1 and only 1 Mandatory to 0 to Many Optional
a User can have 0 or many bookings but a booking can only be tied to 1 User, cannot exist without a User
Inventory
PK id int
FK user_id int
description TextField
item CharField
quantity PositiveIntegerField
User
PK id int
date_joined DateTimeField
is_active BooleanField
is_staff BooleanField
is_superuser BooleanField
password Charfield
username Charfield
FK booking_id int
FK inventory_id int
Booking
PK id int
FK user_id int
completed BooleanField
contact CharField
details TextField
end_time DateTimeField
materials_cost DecimalField
materials_needed TextField
phone_numnber Charfield
quote DecimalField
start_time DateTimeField
title CharField

Figure 29 : Small Scale Business Management Application Entity Relation Diagram

4 IMPLEMENTATION

The application was designed to be built using a three tiered architecture consisting of a presentation tier, logic tier and data tier.
The decision was made that python would be written in majority python, this decision for a three tiered python focused application lead to the selection of Django as a web framework as its use of the Model View Template (MVT) architecture and default SQLite database made it an ideal choice for what was needed during development.
The development itself was carried out in Microsoft Visual Studio Code

4.1 Integrated Development Environment (IDE)

Visual Studio Code is a free open source and versatile integrated development environment created by Microsoft.
Visual Studio Code is designed to be lightweight while providing extensive features designed to streamline the development process, leading it to be a popular option for developers.
Visual Studio Code comes with a code editor which supports syntax highlighting and code formatting for a large number of programming languages, which streamlines the development of applications consisting of multiple languages. IntelliSense is also a feature included which provides code completion, parameter information and quick info on aspects of code such as functions and variables. Also included is an integrated terminal which allows the use of various shells such as PowerShell, Command Prompt, allowing for the execution of commands, error detection and due to Visual Studio Codes integration with Git version control commands can also me executed via the terminal allow for the staging, committing and pushing of code changes to repositories.
Visual Studio Code also offers a large library of extensions and plugins which gives developers the ability to customise their environment such as the ability to add extensions for additional programming language support for those not covered by default such as PHP and Ruby, code snippets extensions which provide prebuilt code templates allowing for increased efficiency when using common code structures.
Formater extensions can also be implemented to help code consistency with the use of style rules and best practises

4.2 The Project

The project is a Python focus application build using the Django framework which is a high level Python web framework that encourages rapid development with clean, pragmatic design utilising the Model View Template (MVT) architectural pattern to separate the given application into three interconnected components, the Model, the View and the Template.

4.3 Django

Django is a free and open-source web framework based on Python that utilises the Model View Template architectural pattern. The Django Software Foundation (DSF) is an independent non-profit organisation registered in the United States which is responsible for its maintenance.
The goal of Django is to simplify the development of database driven web applications. The framework promotes the reusability of components and their ease of integration due to low coupling, quick development and the principle of "Don't Repeat Yourself" (DRY) software development.
Python is used throughout the framework, such as for settings, files and data models. Django additionally offers an administrator interface which allows for the creating, reading, updating and deletion operations by developers to manage an application's data in user friendly interface.

4.4 Model View Template (MVT) Architecture

The Model View Template (MVT) is an architectural pattern used by the Django framework.
The MVT pattern separates an application into three interconnected components, the model, the view and the template.
The Model is responsible for maintaining an applications database and acts as an interface for the data.
The View is responsible for the user interaction and requests and the business logic an the application

The Template is responsible for the presentation of an application and manages the user interface.

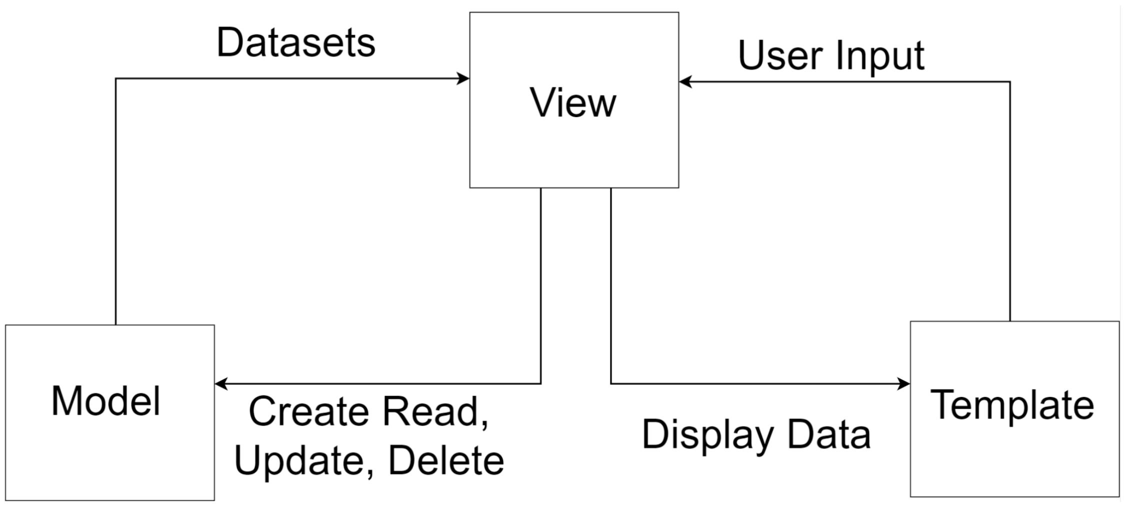


Figure 30 : Model View Template Architecture

4.5 Model View Template (MVT) Components

The components of the MVT architecture are:

- The Model, this is responsible for managing and defining the structure of an applications database and acting as an abstraction layer between the application and the database and allows for the query and manipulation of data without needing to compose SQL statements. In django a model class using python re[resents the database schema with each class attribute acting as a field in the database table. Relationships can be established between various tables in the database with the use of Foreign Keys to create associations between different data entities and allow for a wider scope on data querying and manipulation
- The View, this is responsible for handling user requests, data processing and returning responses to the user based on the user requests. In regards to user requests the View can be used to create specific Python functions or classes to handle user requests and return an appropriate response to the user through interactions with the Model and Template. The View is crucial in implementing the business logic of an application and controlling data flow between the Model and Template.

- The Template, this is responsible for the structure and presentation of the application and the data returned to the user. Templates are HTML files used to separate the presentation structure and the business logic of an application. The HTML is used to define the structure of an applications web page(s), also the use of Django’s own template language allows for the use of dynamic content and data into the HTML. The Template also make use of inheritance to allow the use of a base template with a common structure which can then be extended to be use my for other webpages in the application and then customising it base template based on the need required for other webpages of the application, this promotes code reusability and aids in the maintenance of the application presentation.

4.6 Model View Temple (MVT) Interactions

The Model View Template architecture allows the Model, View and template to work in conjunction to handle user request, data management and processing and the returning of responses to the user.

The interactions of there three components can be outlined as such:

- The View can take in user requests and based on the request then interact with the Model if required.
- The Model then based on the request from the View will read, update or delete data.
- Once the View has processed the data from the Model, it will then use an appropriate Template based on the request and insert data again based on the users request into placeholders in the HTML file of the Template. which is then rendered to the user.

4.7 Benefits of Model View Template (MVT) Architecture

The Model View template architecture offer many benefits for the development of web applications such as:

- The Separation of Concerns
  - Due to MVT architecture separation of data management (Model), business logic (View) and presentation layer (Template) into three distinct parts this allows for more organised and maintainable code which can aid development and help in modifying, extending or adding additional functionality to the application. This separation of concerns also leads to a containment area if issue arises in one component it has minimal impact on the other components

- Quick Development
  - There is a large suite of built-in tools and features available to developers such as authentication system, admin interface, generic class based views, form handling which can be used or built upon and also the ability to inherit and extend Templates leading to reduced time and effort needed to develop common web applications. Due to the number of features and tools available it can be described as a “Batteries Included” architecture

- Scalability
  - Due to the MVT architecture and the separation of concerns this leads to increased scalability as each component can be developed, tested and optimised independently. If and when the application grows only the components needed for growth can be improved upon and remaining components can be untouched if improvements are not required of them.

- Flexibility
  - The MVT architecture aids in giving developers reliability in their applications design and development. application features can be removed or integrated as needed with greater ease and the ability to reuse features and apps

4.8 Design implementation

The system was designed with the aid of a free open source graph drawing software which can be used to create diagrams, charts, graphs, wireframe and UML diagrams.

4.9 Payment Processor Integration

The implementation of a custom payment processor to be used for user registration was not in the scope of this project, however I did intend to implement this feature with the use of a 3rd party application.
The application chosen was provided by Stripe.
Stripe facilitates online payment processing and commerce solutions for internet business of all sizes and also assists in fraud detection and reduced friction at checkout. Stripe supplies a low-code payment integration that creates a customizable payment page so users can easily implement and collect payments on desktop and mobile devices.

4.10 Matplotlibs

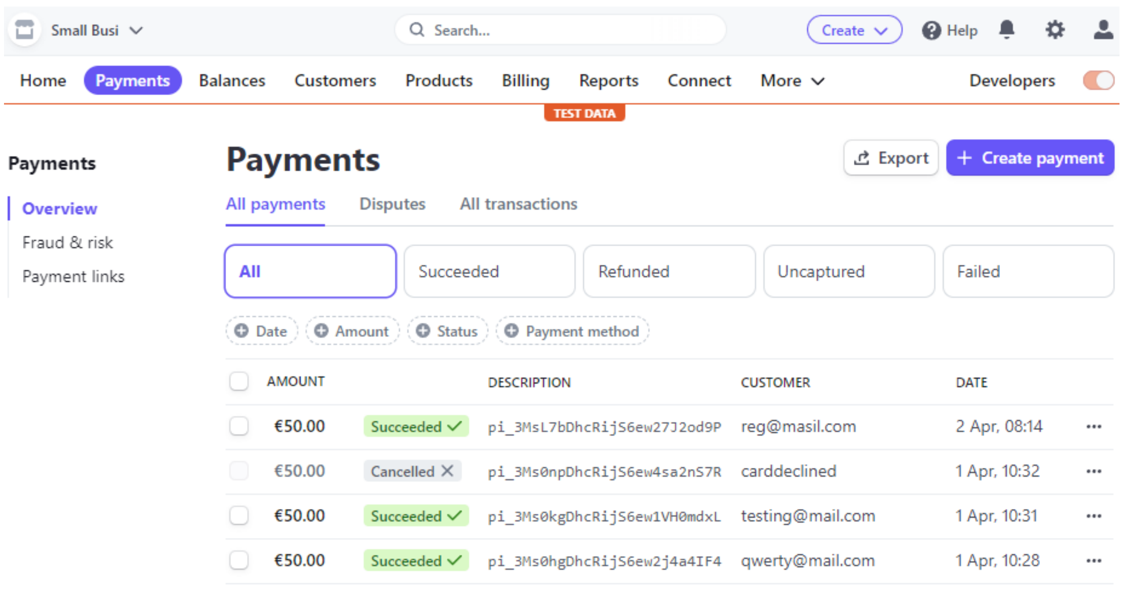
The finance component of the system leveraged the use of Matplotlib in order to give users a simple and clear representation of a user's finances.
Matplotlib is an extensive library used for the creation of static, animated and interactive visualisations in Python to allow for the visualisation of data in the form of bar charts, line pots, scatter plots, 3D plots among others.

5 TESTING

Testing was carried out extensively during development to ensure the application was looking and responding as intended and if this was not the case corrective action could be taken.
Each section of the application was tested independently and also further testing was carried out as components were integrated and testing the application as a whole with the use of regression and unit testing.

There were slight changes made to the application as a result of testing. I initially wanted to have a model appear when a user clicked a inventory item or calendar booking that would display the selected information and allow the user to edit the data and save updates, however I ran into issue with data for the selected item not been displayed for the selected item, I was able to pass this data in the applications current iteration so believed it was an issue in how data is passed when a form is in a model as this issue does not occur when a model is not used. I was unable to resolve this issue due to time constraints but I believe the current iteration provides a better user experience as the original design would have a user select an Inventory item, this would bring up a model containing a form and then id the user wished to delete the data another model would be displayed to confirm deletion and I believe this would have given the editing and deleting functionally a messy and clutter appearance.

The integration of Stripe was also tested to ensure it had been correctly implemented with the project and that test payments were processed and payments were cancelled if there was an issue with a card, this was carried out using test card provided by Stripe and when checking my Stripe account I could see payment success and failures as expected based on the cards used.



AMOUNT		DESCRIPTION	CUSTOMER	DATE
€50.00	Succeeded ✓	pi_3MsL7bDhcRijS6ew27J2od9P	reg@masil.com	2 Apr, 08:14
€50.00	Cancelled ✕	pi_3Ms0npDhcRijS6ew4sa2nS7R	carddeclined	1 Apr, 10:32
€50.00	Succeeded ✓	pi_3Ms0kgDhcRijS6ew1VH0mdxL	testing@mail.com	1 Apr, 10:31
€50.00	Succeeded ✓	pi_3Ms0hgDhcRijS6ew2j4a4IF4	qwerty@mail.com	1 Apr, 10:28

Figure 31: Stripe Payment Processing Status

The following are test cases for the application written using the Gherkin syntax. The Gherkin syntax uses a set of special keywords such as "Given", "When", "Then", "And" to give a structured, plain text format that allows for the creation of clear and concise test cases that can be understood by technical and non technical readers alike.

Feature: Registration Testing

Scenario: Successful Registration of a User

GIVEN the application login page has loaded
WHEN the Subscribe and Register link is clicked
THEN the Payment page is loaded
WHEN the Checkout button is clicked
THEN redirected to the Stripe Hosted Payment page
AND necessary info is used in the required fields
WHEN the Pay button is clicked
THEN redirected to the Register page
AND necessary info is used in the required fields
WHEN the Complete your Registration button is clicked
THEN user is registered
AND redirected to the Login page

Scenario: Unsuccessful Registration of a User

GIVEN the application login page has loaded

WHEN the Subscribe and Register link is clicked

THEN the Payment page is loaded

WHEN the Checkout button is clicked

THEN redirected to the Stripe Hosted Payment page

AND necessary info is used in the required fields

WHEN the Pay button is clicked

THEN redirected to the Register page

AND necessary info is used in the required fields

AND input is not accepted

THEN user is not registered

Feature: Login Testing

Scenario: Successful Login of a User

GIVEN the application Login page has loaded

WHEN a registered user enters their login credentials

THEN the user is logged into the application

AND redirected to the application Home page

Scenario: Unsuccessful Login of a User

GIVEN the application Login page has loaded

WHEN a registered user enters incorrect credentials for their account

THEN a warning message is displayed advising of the use of incorrect login credentials

AND the user is not logged in

Feature: Inventory Testing

Scenario: User Views Inventory Page

GIVEN a User has logged into their account

WHEN a User clicks the Inventory button in the Navbar

THEN the User is redirected to the Inventory page

Scenario: User Adds a New Item To Their Inventory

GIVEN a User has logged into their account

AND the User is on the Inventory page

WHEN a User clicks the Add New Item button

THEN the User is redirected to the Add new Inventory Page

WHEN a User enters in the required Inventory item details

AND clicks the Save button

THEN the new Inventory item is saved to the Inventory database

AND the user is redirected to the Inventory Page

AND the new Inventory item has been added to the page

Scenario: User Views an Inventory Item

GIVEN a User has logged into their account

AND the User is on the Inventory page

WHEN a User clicks the Update button next to an Inventory item

THEN the user is redirected to the Inventory edit page

AND the details for the selected Inventory Item is displayed

Scenario: User Updates an Inventory Item

GIVEN a User has logged into their account

AND the User is on the Inventory page

WHEN a User clicks the Update button next to an Inventory item

THEN the user is redirected to the Inventory edit page

AND the details for the selected Inventory Item is displayed

THEN the User edits the Inventory items details

AND the User clicks the Save button

THEN the Inventory item is updated in the Inventory database

AND the User is redirected to the Inventory page

AND the update is visible on the Inventory page

Scenario: User Deletes an Inventory Item

GIVEN a User has logged into their account

AND the User is on the Inventory page

WHEN a User clicks the Delete button next to an Inventory item

THEN a confirmation modal is displayed

WHEN a User clicks the Confirm button

THEN the confirmation modal closes

AND the Inventory item is deleted from the Inventory database

AND the Inventory item is removed from the Inventory page

Feature: Calendar Testing

Scenario: User Views Calendar page

GIVEN a User has logged into their account

WHEN a User clicks the Calendar button in the Navbar

THEN the User is redirected to the Calendar page

Scenario: User Creates a Booking

GIVEN a User has logged into their account

AND the User is on the Calendar Page

WHEN a User clicks the New Booking button

THEN the User is redirected to the create new booking page

WHEN a User enters all the required info for a new booking

AND clicks the Save button

THEN the new booking is saved to the Calendar database

AND the user is redirected to the Calendar page

AND then new booking is added to the Calendar

Scenario: User Views a Booking

GIVEN a User has logged into their account

AND the User is on the Calendar Page

WHEN a User clicks a Booking on the Calendar

THEN the User is redirected to the booking edit page

AND the details for the Booking are displayed

Acceptance Criteria - User Updates a Booking

GIVEN a User has logged into their account

AND the User is on the Calendar Page

WHEN the User clicks a Booking on the Calendar

THEN the User is redirected to the booking edit page

AND the details for the Booking are displayed

THEN the User edits details of the Booking

AND clicks the Save button

THEN the Booking is updated in the Calendar database

AND the User is redirected to the Calendar Page

Scenario: User Deletes a Booking

GIVEN a User has logged into their account

AND the User is on the Calendar Page

WHEN the User clicks a Booking on the Calendar

THEN the User is redirected to the booking edit page

AND the details for the Booking are displayed

WHEN the User clicks the Delete button

THEN a confirm modal is displayed

WHEN a User clicks the Confirm button

THEN the confirmation modal closes

THEN the Booking is deleted from the Calendar database

AND the Booking is removed from the Calendar page

THEN the User is redirected to the Calendar page

Feature: Finance Testing

Scenario: User Views Finance page

GIVEN a User has logged into their account

WHEN the User clicks the Finance button in the Navbar

THEN the User is redirected to Finance page

Scenario: Finance page Bar Chart Responds to Booking Data

GIVEN a User has logged into their account

WHEN a User creates a booking and supplies a quote and material costs

THEN the User should see the data reflected in the Finance Page Bar Chart

Feature: Logout Testing

Scenario: User Logs Out Of The Application

GIVEN a User has logged into their account

WHEN the User clicks the Logout button in the Navbar

THEN the User is redirected to Logout page

AND the User is Logged out of the application

6 CONCLUSION

Through the development of the application I gained a greater appreciation of the planning and requirements gathering stages of development and how this stage helps in solidifying the needs, look, functionality and how the system goes about achieving the desired functionality leading to a clearer and more defined development process.
During development issues may have been foreseen with further planning and solutions could be researched before encountering them, such as an issues I encountered with the use of forms used within models. As I had used forms in other applications previously I was confident in my ability to implement them in the application, which did work, however when I attempted to use a model to display and allow a user to edit the data displayed I discovered data had not been passed to the form. I found this difficult to debug as I there was not error codes or any indication of an issue by the system, but due to time constraints I had to abandon this design and leave the application in it current form, which I believe provides a nicer user experience as I had originally envisioned and model displaying form data to be edited if needed and if the data was to be deleted another model would have been displayed on top of the original and I believe this would look cluttered.
Another issue or room for enhancement is in the Calendar component of the application.
Currently there is no highlighting of booked jobs on the calendar which is for single booking is not much of an issue, but the issue occurs if there is a booking that spans multiple days which due to the audience the application is designed for it is very likely to occur.. When a booking is made a title is placed in the date cell for the start of the booking but no indication if the job spans multiple day, I attempted to add an indication of this by a clourled bar for each booking on the calendar that would highlight on the calendar the start date to end date cell on the calendar but I ran into issue with overlapping bars and bars for bookings extending out of the bounds of the calendar if a booking carried from one week to the next. I had to leave this as I wanted to complete what I deemed more beneficial aspects of the application in the hope that I would have time to return and properly implement this feature, which unfortunately I did not

Another issue encountered late in development was the user registration aspect. Here a user that wishes to register is directed to a payments page where they filling in their card details, the card is then processed and charged and if successful the user is then directed to the registration page, however I didn't not account for a user been able to bypass the payment process by access the registration page directly by using its URL, I would have been unable to restrict access to this page like it had for the rest of the site for unregistered users, as an unregistered user would need access to this page inorder to complete registration. Again due to time restraints I was unable to fully research and implement a solution to this, I did invision that when a payment is successful at token could be passed from the payment processor (Stripe) that would attach to the users web session and this token would be checked or if a user attempted to access the registration page directly and if not present the user would be blocked access until payment was processed and the token was attached to their user session, but due to time constraints I was unable to research if this would be a possible or a viable solution.
Another issue relating to payment is the use of Stripes API keys to use there payment processing, for security reason they should be stored in the setting.py file of the application and the Secret key used a environment variables, however I ran into issue with this, I was able to set the set the Secret Key and confirm this by printing it to the terminals, but it seems there was an issue it it been read when attempting access Stripes checkout page, I was unsure on how to resolve this as the error message that it could not be read contradicted the fact that it had been set and could be confirmed by printing to the terminal. To work around this I need to include the Secret Key in the Payment components views.py file which is bad practice from a security standpoint

With the above taken into account I am still overall happy with the final outcome of the application and intend to research this issue in the hopes of improving the application.
Another aspect I had initially planned for the application was that the frontend would be handled by React instead of by Djando which would help in the separation of concerns but mainly due to my interest been more in the frontend aspect of web development, at the time I had only a very brief experience using React building small single app and had not attempted to integrate React and Django. I decided against using them both due to my lack of experience and therefore lack of foresight of possible issues I may encounter but this is something I would like to implement at a later stage.

BIBLIOGRAPHY

https://docs.djangoproject.com/en/4.1/

https://www.w3resource.com/python/module/calendar/index.php

https://epydoc.sourceforge.net/stdlib/calendar.HTMLCalendar-class.html

https://www.geeksforgeeks.org/python-calendar-module/

https://stackoverflow.com/

https://docs.djangoproject.com/en/4.1/ref/forms/widgets/

https://getbootstrap.com/docs/4.1/components/modal/

https://www.w3schools.com/howto/howto_css_delete_modal.asp

https://linuxhint.com/bootstrap-delete-confirmation-modal/

https://matplotlib.org/stable/users/explain/backends.html

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

https://www.w3docs.com/snippets/html/how-to-display-base64-images-in-html.html

https://www.w3schools.com/django/ref_tags_now.php

https://docs.djangoproject.com/en/4.1/topics/db/managers/

https://en.wikipedia.org/wiki/Django_(web_framework)

https://code.visualstudio.com/docs

https://code.visualstudio.com/docs/python/tutorial-django

https://docs.djangoproject.com/en/4.1/ref/contrib/auth/

https://www.askpython.com/django/django-mvt-architecture

https://sqlite.org/index.html

https://app.diagrams.net/

https://cucumber.io/docs/gherkin/reference/

https://stripe.com/ie

APPENDIX